



Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

1993-12

Multicast communication with guaranteed quality of service

Boyer, Eric B.

Monterey, California. Naval Postgraduate School

<http://hdl.handle.net/10945/39663>

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>

AD-A277 650



NAVAL POSTGRADUATE SCHOOL
Monterey, California

②

S **DTIC**
ELECTE
F **D**
APR 04 1994



94-09969



103P8

THESIS

**MULTICAST COMMUNICATION
WITH
GUARANTEED QUALITY OF SERVICE**

by

Eric B. Boyer

December 1993

Thesis Advisor:
Second Reader:

Shridhar B. Shukla
Gilbert M. Lundy

Approved for public release; distribution is unlimited.

DTIC QUALITY INSPECTED 1

94 4 1 041

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 16 December 1993.	3. REPORT TYPE AND DATES COVERED Master's and Engineer's Thesis		
4. TITLE AND SUBTITLE MULTICAST COMMUNICATION WITH GUARANTEED QUALITY OF SERVICE		5. FUNDING NUMBERS		
6. AUTHOR(S) Eric B. Boyer				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE A		
13. ABSTRACT (maximum 200 words) In this thesis, we address the problem of constructing multicast data distribution trees with guaranteed quality of service (QoS) for supporting multiparty interactions. We present an approach that integrates reservation with tree construction to facilitate a guaranteed quality of service. The proposed approach is based on the use of information about participants registered before the interaction starts. We first identify the design goals for multicast tree construction with minimum QoS requirements. We then describe a protocol to locate a set of distribution centers for an interaction that depends upon the current load distribution, locations of the participants, and their QoS requirements. The protocol sets up a suitable number of center-specific trees for the interaction transparently. We compare the quality of the resulting trees on large, hypothetical networks with that of sender-specific and Steiner trees. Our results show that center-specific trees, built around the centers located by our approach, reserve fewer resources than sender-specific trees even for a significant number of simultaneous senders while sacrificing minimally in the average delay faced by each receiver.				
14. SUBJECT TERMS Multicast, Quality of Service, Wide Area Network, Critical Set of Participants, Scheduling Register			15. NUMBER OF PAGES 103	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)

Prescribed by ANSI Std. Z39-18

Approved for public release; distribution is unlimited.

**MULTICAST COMMUNICATION WITH
GUARANTEED QUALITY OF SERVICE**

by

Eric Blaine Boyer
Lieutenant, United States Navy
B.S. Systems Engineering, United States Naval Academy, 1986

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN ELECTRICAL ENGINEERING
and
ELECTRICAL ENGINEER**

from the

NAVAL POSTGRADUATE SCHOOL


December, 1993


Author:


Eric B. Boyer

Approved By:


Shridhar B. Shukla, Thesis Advisor


Gilbert M. Lundy, Second Reader


Michael A. Morgan, Chairman,
Department of Electrical and Computer Engineering


Richard S. Elster, Dean of Instruction

ABSTRACT

In this thesis, we address the problem of constructing multicast data distribution trees with guaranteed quality of service (QoS) for supporting multiparty interactions. We present an approach that integrates reservation with tree construction to facilitate a guaranteed quality of service. The proposed approach is based on the use of information about participants registered before the interaction starts. We first identify the design goals for multicast tree construction with minimum QoS requirements. We then describe a protocol to locate a set of distribution centers for an interaction that depends upon the current load distribution, locations of the participants, and their QoS requirements. The protocol sets up a suitable number of center-specific trees for the interaction transparently. We compare the quality of the resulting trees on large, hypothetical networks with that of sender-specific and Steiner trees. Our results show that center-specific trees, built around the centers located by our approach, reserve fewer resources than sender-specific trees even for a significant number of simultaneous senders while sacrificing minimally in the average delay faced by each receiver.

Accession For	
NTIS	CRA&I
DTIC	TAS
Unannounced	
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

TABLE OF CONTENTS

I.	Introduction	1
	A. An Integrated Approach based on Critical Participants	3
	B. QoS Parameters of Multicast Multimedia Communication	5
	1. QoS Requirements of Point-to-point Channels	6
	2. A Virtual Classroom	7
	3. A Teleconference	7
	4. A Panel Discussion	8
	5. A Distributed Interactive Simulation	9
	6. A Command and Control Scenario	9
	C. Scope and Contributions	10
	D. Thesis Organization	11
II.	Design Goals	12
III.	Comments on Existing Approaches	16
	A. Core-based Trees	16
	B. Protocol Independent Multicast	17
	C. Distance Vector Multicast Routing Protocol	18
	D. MOSPF	20
	E. Comparison against Design Goals	21
IV.	General Approach	22
	A. Assumptions	22
	B. Use of <i>a priori</i> Information About the Participants	25
	C. Locating a Set of Distribution Centers	26
	1. Selecting the Location of a Distribution Center	27

2.	Illustration of the Selection of a Center	29
3.	Functions of the SR	29
D.	Tree Construction	32
E.	Reserved Routing	34
V.	Protocols for Locating the Distribution Centers	38
A.	Participant Registration Protocol	39
B.	Center Selection Protocol	41
VI.	Performance Evaluation	45
A.	Generation of the Network Topology	45
B.	Simulation Results	47
C.	Qualitative Comparison With PIM	50
VII.	Techniques to Guarantee QoS	51
A.	Stop and Go Queueing	51
B.	Tenet Group	52
C.	Predictive Service	53
D.	RSVP	54
VIII.	Concluding Remarks	55
A.	Suggestions for Future Research	55
	APPENDIX A. PROGRAM CODE	58
	LIST OF REFERENCES	87
	BIBLIOGRAPHY	90
	INITIAL DISTRIBUTION LIST	94

LIST OF TABLES

1.1	QOS PARAMETERS FOR POINT-TO-POINT CHANNELS	6
3.1	EXISTING APPROACHES VS. DESIGN GOALS	21

LIST OF FIGURES

4.1	Illustration of assumptions	24
4.2	An Example Network	30
4.3	Selection Tree for the Example Network	31
5.1	Algorithm for Determining the Selection Hierarchy	41
5.2	Determining the Bye Positions in the First Phase	42
6.1	Tree cost for $\lambda = 4$ and 16 CP's	48
6.2	Tree cost for $\lambda = 3$ and 16 CP's	49

ACKNOWLEDGMENTS

I would like to give special thanks to Dr. Shridhar B. Shukla for spending countless hours discussing papers and pondering questions.

I. Introduction

The integrated packet switched networks of the future are expected to provide users with the ability to engage in various types of multiparty interactions. Some examples of these are teleconferencing, virtual classroom, remote panel discussion, remote telemetry and reconnaissance, distributed simulation/test environment, and virtual cafe. All these interactions benefit from a network-level multicast with a guaranteed Quality of Service (QoS).

Network-level multicasting refers to reduction in the amount of traffic in the network for implementing point-to-multipoint communication when compared to multiple unicasts. Wherever the associated unicasts share a path, a single packet is sent. Where the paths of the unicast diverge, duplicates of the packet are created and routed down each of the ensuing paths. The multicast packet is addressed to a group address instead of an individual address. This requires the network to know the members of the group and their location. In order to prevent the inefficiency resulting from a broadcast that would flood the network to get to destinations that correspond to members, a multicast tree is formed. This tree directs the packets toward the members and avoids routing the packet along any path that does not lead to a member of the group. Use of some such technique for tree-construction is essential if the efficiency gained in multicasting is not to be counteracted by the waste of resources in a broadcast. [Ref. 1]

Current network-level multicast techniques are aimed at providing a *best-effort* delivery, on lines similar to the UDP datagram delivery. In the presence of congestion (lack of resources) and/or failures, packets may not be delivered. A multicast with a

guaranteed QoS refers to provision of guarantees at least as far as the availability of network resources required for delivery of multicast packets is concerned.

There are two major approaches to multicast tree construction: center-specific trees (CST) ¹ [Ref. 2] and sender-specific trees (SST) [Ref. 3, 4]. A single tree, rooted at some center (router) is shared by all senders to a group in the former approach, while each sender builds a separate tree rooted at itself in the latter. The advantages of a CST are that network routers must maintain interface state only for a single tree per group regardless of the number of senders to it, and in case of multiple simultaneous senders requiring reservations, the total amount of resources is less than SSTs. The drawbacks of a CST, as proposed in [Ref. 2], are that, for large and/or widely spread groups, certain backbone links may become a bottleneck, and in case of groups with all members being senders requiring QoS, network resources may be utilized non-uniformly resulting in traffic-concentration [Ref. 5]. Another drawback of this technique is that, in the existing literature, there is no proposal for quickly locating a core to build a CST from. The center-specific tree construction approaches have proposed that the centers be selected administratively. This is likely to prevent congestion of traffic in the network by locating the center based on the group members' locations. ²

The advantages of source-specific trees are that it is scalable and efficient in case of a large number of simultaneous senders (even if they require guaranteed QoS). The volume of traffic carried by each tree remains the same regardless of the number of senders. It is also argued that the source-specific approach permits the highly desirable flexibility in providing receiver-initiated reservation to guarantee QoS

¹We avoid the use of center-based trees to prevent any confusion between its short-form, CBT, which is popularly used for core-based trees [Ref. 2]. The term is also more appropriate for the proposed approach based on distribution centers.

²We view a Steiner tree to be a special case of a CST because all the proposed algorithms for Steiner trees are based on a pre-selected root, which we regard as a center. We focus not on the tree quality in our classification but on the protocol for tree construction.

[Ref. 6]. Its main disadvantage is that, when the number of simultaneous senders is likely to be small compared to the total number requiring a guaranteed QoS, excessive reservation of network resources occurs. Basically, reservation will occur along every tree although, not all trees will carry traffic at the same time.

A general drawback of both the above approaches to tree construction is that they are mainly concerned with routing of multicast data and have not addressed the techniques to provide guaranteed QoS. The ability to provide a guaranteed quality of service depends on the ability to reserve resources. Since resources are consumed along the routes taken by multicast traffic, it is natural to expect that, in a technique that guarantees QoS, routing be coupled with reservation. There are a great number of ways to accomplish this goal. The different requirements for each application will lend themselves to different ideal ways to handle the problem. To attempt to find the best median for all possible applications is very extensive and furthermore hampered by the impossible task of knowing all possible applications.

To summarize the above discussion, we observe that efficient wide-area multiparty interactions with QoS requirements need a tree construction technique that is scalable in terms of the number of participants and geographical distribution of members, uses network resources efficiently, integrates reservations, and requires low tree-state maintenance overhead, and places minimal administrative burden. We address the problem of providing such a technique for tree construction. Our approach is targeted at multiparty interactions requiring reservations and for which some *a priori* information about some participants is available.

A. An Integrated Approach based on Critical Participants

With each multiparty interaction mentioned above, a critical set of participants can be identified. For example, in a teleconference, any quorum of participants

permits the meeting to perform its function. Every attendee is a potential sender and a receiver. In a virtual classroom, the lecturer is the only critical participant. Any number of students, permitted by the room size, may be present. The lecturer is a sender as well as a receiver and the students are only receivers.³ However, their presence is not essential for the class to go on. In a panel discussion, all the panel members form the set required for the discussion to begin. Any number of listeners may be present. In order to keep the panel effective, it should at least be ensured that all the panelists receive every other panelist with a guaranteed QoS. In a remote telemetry application, some data collector must exist for the sensors to send data to. This collector forms the critical set. In a distributed simulation/test environment, all the entities form the set of critical participants. In a virtual cafe, the critical set is formed dynamically by one or more people who acknowledge each other's desire to interact and try to establish communication.

It is important to note that the primary attributes of a critical participant from the network's perspective are: its location, the interaction it wants to participate in, and its frequency of sourcing data relative to other participants. We use these attributes as an aide to determine one or more central network locations for critical participants based on which one or more efficient CSTs can be built.

Relying on critical participants also solves the following problem associated with tree construction for the participants of a multiparty interaction. Finding the center for a multicast shared tree requires an exhaustive search. In addition, when the membership is dynamic, the ideal center will also move. It may be desirable to have a relatively static tree, that is, one in which the root does not move. For this case, the tree should not rely on temporary members for determining the root of the tree.

³Of course, occasionally a student may become a sender. However, it is unrealistic to expect the network to keep the resources reserved in view of the low frequency with which a student may send. In any case, the teacher is expected to repeat the student's remarks for the benefit of the rest of the class.

Those members that are long-lived and possibly produce a relatively large amount of traffic to the group should carry the most weight. As we have seen, these members will be known *a priori*. In addition, if the number of members used to determine the root of the tree is small, then the number of calculations to determine the center can be greatly reduced.

Using critical participants also provides a clean approach for resource reservation. While it is impossible to anticipate the needs of dynamic members, the tree for the critical participants can be constructed based on whether resources are available or not. The network, with the knowledge of the critical participants, can form a tree with the required resources well-before the interaction begins. This follows the successful model of day-to-day life in which pre-planned activities can block resources as soon as the planning completes and resources are made available for reservation (community halls, airline seats, opera seats, etc.) whereas unplanned activities run the risk of resource unavailability. This essentially integrates routing with reservation.

B. QoS Parameters of Multicast Multimedia Communication

We now briefly describe the QoS requirements of multiparty interactions over packet-switched networks that have been understood so far by the community. Multimedia data may correspond to audio packets, video packets, slowly changing graphics, and normal bursty data. Each has its unique requirements in a point-to-point setting. The problem of setting up a multiparty interaction is to satisfy these requirements even in point-to-multipoint cases.

TABLE 1.1: QOS PARAMETERS FOR POINT-TO-POINT CHANNELS

Type	Delay	Loss Prob.	Bandwidth	Conditions
Audio	300 msec	800 bits (cons.)	64 Kbps	PCM, 8 bit samples
Video	300 msec	4.16 Kbits per frame	1.34Mbps	0.25 screen, byte stream, 20:1 min. comp.
Graphics	n/a	n/a	208 Kbps	full screen window, 0.5 sec update, min 20:1 comp.
Simulation PDU	100-300 msec	0	Application dependent	
Sensor data	Application dependent	Application	Application dependent	

1. QoS Requirements of Point-to-point Channels

The QoS required by a point-to-point channel has been primarily characterized by the maximum permissible end-to-end delay, end-to-end jitter, and loss probability [Ref. 7]. For packet-switched networks, however, it has been recently established that end-to-end delay and jitter can be bounded by ensuring that a channel receives at least a certain amount of average bandwidth at each intermediate router [Ref. 8]. Therefore, providing a guaranteed QoS point-to-point channel amounts to reservation of appropriate bandwidth at each intermediate router. Also, by providing an appropriate amount of buffering, the end-to-end jitter requirement can be met. Typical QoS parameters for various types of channels are summarized in Table 1.1 [Ref. 9, 10].

A number of these sources, with or without a need for synchronization among them, may be present in an interaction. Types of interaction differ based on the number of and the interrelationships between these sources. Based on the numbers above, we describe five sample interactions, *a virtual classroom*, *a teleconference*,

a panel discussion, and a distributed interactive simulation below.

2. A Virtual Classroom

In this interaction, the teacher represents the CP and the students are the receivers who may join and leave the interaction at any time. Depending upon the operational details of the classroom, only the student audio may be carried back either *only* to the instructor or to all the students along with the instructor. Or, both the audio and video from each student may be carried to either *only* the instructor or to all the students along with the instructor. In either case, it does not seem necessary that reservations are needed from the students to the teacher for more than one student at a time.

Thus, assuming that *only* the student audio is carried *only* to the instructor, a virtual classroom will require one video (the instructor), one graphics (the board or the screen), and one audio channel from the teacher to all the students. One audio channel will be needed from all the students to the instructor. It is assumed that the students ask questions only when the instructor solicits them and that the students are polite enough to not monopolize the audio channel. Synchronization will be required among the channels originating at the teacher. A SST rooted at the teacher and a CST rooted at an appropriate router shared by all the students will be required.

3. A Teleconference

A teleconference is much like a virtual classroom in terms of the number of simultaneous senders. The number of receivers is likely to be restricted to the set of conference participants. Every participant is a potential sender and is always a receiver. A meeting is usually a planned interaction, and therefore, all the participants

locations are expected to be known *a priori*. It is assumed that the participants are polite enough that only one speaks at a time. It is likely that more than one are to be permitted to speak at one time. However, we cannot find a reason for requiring more than one speaker at a time.

In terms of the number of channels, each participant is a potential sender to a video channel, a graphics channel, and an audio channel. With the above restriction, the graphics and audio channels could be sent over a shared tree. Each video channel could be sent over a SST to enable all participants to have a window for all other participants.

4. A Panel Discussion

A panel discussion has the same characteristics as a teleconference combined with a panel discussion. There are only a few potential senders, the CPs, and a large number of receivers who may join and leave the interaction at any time. The panelists' locations are known *a priori* and are unlikely to leave in the middle of the interaction. The receivers' membership may be short-lived.

A video channel originating at each of the panelists must be distributed to all the receivers (including the panelists). An audio and a graphics channel must also reach all the receivers and the panelists. Depending upon the operational details of the panel discussion, a receiver may receive the live video of every panelist's face along with the speaker's audio and graphics channel or may see a still graphic image of each and a live video of only the current speaker. The audio and graphics each receiver gets must be synchronized with the video received. Assuming that there is wastage in transmitting live video of a face and the latter option is selected, the current speaker's audio, video, and graphic channel is to be distributed to all the receivers as well as panelists. Thus, a CST can be constructed for each of the channels with

the core located with respect to the panelists and the receivers receiving multicast traffic from the distribution center (root) of this CST.

5. A Distributed Interactive Simulation

In a distributed interactive simulation, multiple hosts execute simulator programs simultaneously. Each simulator sends protocol data units (PDUs) to all the other simulators so that every simulator's state is made consistent with that of all the others periodically [Ref. 11]. There are strict latency constraints on all the PDUs and the rate of generation of the PDUs from a simulator depends upon its function as well as the events it simulates. In terms of the multicast requirements, a set of DIS applications presents a set of concurrent senders who also receive all the PDUs. Recent DIS experiments have used up to eleven hosts [Ref. 11], but in the near future, this number may grow as the scope of the simulations grows.

In this multiparty interaction, each simulator host is a CP that needs to construct a SST rooted at itself with the QoS determined by the PDU constraints. In this case, there is no benefit in sharing a tree.

6. A Command and Control Scenario

In a command and control scenario as envisioned in COPENICUS [Ref. 12], two of the many infrastructure related applications are *information-pull* and *creation of a consistent tactical picture ashore and afloat*.

In the *information-pull* application, the commander-in-chief (CINC) will determine the sources from where information is to be gathered. This will result in multiple concurrent sources sending to one or more sites at which the CINC needs the information sent. However, this set of sources is likely to be a subset of a much larger set of information producing sites. Since the total number of possible sources is likely

to be much larger than the number of sources required in any single information-pull operation, a shared CST rooted at an appropriately located center will be more efficient.

In the *creation of a consistent tactical picture*, every source contributing to such a picture must update all the sites that maintain the picture. At any time, the sources that actually generate information will be determined by the geographical area in which the relevant events take place. If such events can be anticipated, non-concurrent sources in an area could share CSTs. The centers of these trees act as sources for all the recipients ashore and afloat. SSTs, where the centers of the lower level CSTs become sources, need to be constructed since every lower level CST could have at least one sensor that is active all the time. This leads to a two-level combination of CSTs and SSTs.

C. Scope and Contributions

This thesis addresses the problem of constructing multicast trees with guaranteed QoS that utilize the network resources efficiently. Efficient usage implies load balancing as well as use of minimum amount of resources for any one tree. It focuses on how to achieve an efficient combination of CSTs and SSTs based on the *a priori* information about the participants, locate a center for CSTs transparently to balance the network load, and integrate reservation with routing.

The contributions are as follows:

- A systematic approach, based on the critical participants, is described to select a combination of CSTs and SSTs for an interaction.
- A robust and scalable center location mechanism and protocol that selects a center transparently in a distributed fashion and balances the network reservations is described.

- It is shown, by simulation modeling, that CSTs based on center location as above, provide almost the same delay as SSTs.
- It is shown, by simulation modeling, that CSTs are efficient when the number of concurrent senders is small as compared to the number of participants.
- It is also shown that CSTs with centers located as above compare very well with Steiner trees.

D. Thesis Organization

This thesis is organized as follows. Chapter II describes our design goals and related justifications. Chapter III reviews current approaches by listing the shortcomings of each approach with respect to these design goals. Chapter IV describes our approach in detail and introduces the mechanisms required for implementing it. Chapter V specifies the mechanisms individually and exposes their interrelationships. Chapter VI describes the simulation modeling, performance related data, and analysis for our approach to tree construction. Chapter VII reviews some of the currently proposed schemes for low-level handling of unicast packets with respect to their suitability to guaranteed QoS multicast. The thesis concludes by a summary of the strengths and weaknesses of the proposed approach and the future work required.

II. Design Goals

In this chapter, we describe the desirable features of multicast tree construction when guaranteed QoS is desired.

Use of *a priori* Information About Participants: As seen in the previous chapter, every interaction, except for the virtual cafe type, has some element of planning in it. In fact, typically, considerably more information, such as probable senders' locations, critical receivers, relative rates of sourcing data, etc., is available. It is natural that this information be used in configuring multicast trees.

When such information is used, it adds several desirable features to the quality of trees constructed. For example, the tree topology incorporating the relatively long-lived and important group members remains static. Short-lived members affect the topology only by grafting the appropriate branches. The tree computation can be performed only for a small subset of the important members. The number of distribution centers can be made proportional to the number of concurrent senders. Essentially, an efficient compromise between SST and CST can be reached for the given group.

Automatic Location of Distribution Centers: In a rich network topology, the SSTs for a set of senders are more likely to be different from each other than in a poor network topology. Thus, in a rich topology with many multicast groups, SSTs are likely to balance the network utilization better than CSTs which are all constructed based on the same administratively located center [Ref. 2, 13]. This goal requires that distribution centers must be located for

each specific set of group members. If the current reservation in the network is taken into account while locating such a center, the resulting CST will not only balance the network usage, but will also balance the reserved bandwidth usage. Goal 1 above is complementary to this goal in that the participants' attributes registered according to it must be used to locate the center transparently for a group.

Flexible Selection Between Source-specific and Center-specific Trees: CSTs

and SSTs present a clear trade-off when reservations are required. SSTs reserve less when the number of concurrent senders is large and CSTs reserve less when this number is small. Ideally, the set of senders should be partitioned into subsets in such a way that at least one sender from the subset is active at any time. A separate SST should be constructed for each such set with the senders in the subset sharing a CST. The center of the subset should be the root for the SST. This will minimize the number of reservations in the network. For enabling this efficiency, a flexible selection between SSTs and CSTs must be permitted by the tree construction mechanism.

Integration of Tree Setup with QoS: In any situation that demands guaranteed resource availability, it is imperative that reservations be made as soon as the later of the two events occurs – resources become available for reservation and it is known which resources are to be reserved. This successful model of day-to-day life must be followed in a dynamic environment such as an internet. The length of the interval between the granting of the permission to reserve and the expected start of the interaction could be based on the service charges. At the start of this interval, which resources are needed could be determined by initiating the tree construction mechanism. This mechanism must be sensitive

to which parts of the network are already heavily booked. ToS-related costs need to be used to determine the shortest paths. These actions integrate reservations for guaranteed QoS with tree construction instead of providing them as an afterthought [Ref. 13].

Support of Multiple Routing Protocols: Given that different routing domains in an internet are likely to use different routing protocols, a practical tree construction approach must be compatible with them. It must also not impose any additional state collection overhead of its own. Ideally, it should use some generic measure of network state that any intra-domain routing protocol would collect. The cost of a path between two nodes inside a domain is one such measure.

Minimal Tree State Information: It is expected that multiparty interactions described earlier will lead to sparse as well as locally dense distributions of a large number of members in a group. Membership of senders as well as receivers is expected to be dynamic. A new sender as well as receiver should suffer as small latency as possible. An extreme, yet possible, solution is that every router in the internet maintain state related to every group, and possibly, every sender in each group. Clearly, this solution is not scalable at all. The PIM proposal requires each router aware of the group store state information for each sender of the group [Ref. 13]. On the other hand, the CBT approach requires each router to maintain information for the core of each group [Ref. 2]. We require that minimum possible join latency be achieved while keeping the state information stored by each router minimum.

Minimal Per-packet Processing in the Routers: It is observed that the tree state information maintained by PIM is eliminated in CBT at the expense of

additional per packet processing [Ref. 2]. We require that the proposed protocol keep this overhead in routing multicast traffic low.

Participation of the Senders as well as Receivers in Reservation: Receiver-initiated approach has been suggested as a scalable approach to reservation that is independent of the protocol used for routing and for tree construction [Ref. 6]. This approach requires that the internet paths be symmetric. In an interaction where the set of receivers is relatively dynamic, each new receiver must create reservations for receiving traffic from all the senders. On the other hand, if the senders are relatively long-lived, it should not be required that the entire reserved path from each sender be created for every new receiver. We require that the burden of creating reserved paths be shared flexibly by both the senders as well as receivers.

III. Comments on Existing Approaches

A. Core-based Trees

A current technique that sets up a single shared tree is the Core Based Tree proposal. For this, a router is chosen as the root, or core, of the tree for administrative purposes. New branches of the tree are created at the time of propagation of the join ack message sent by the first on-tree router that receives a join request from a new member. The join request and ack are propagated using unicast methods. A multicast packet from the sender propagates towards the core (which is the packet destination address) using normal unicast. When it hits an on-tree router, its destination address is replaced by the multicast group address found in the packet header and is then disseminated as a multicast message on the tree.

One nice feature of CBT is that, for new senders as well as non-receiving senders, any unicast algorithm works. The main advantages of a core are as follows. Firstly, it provides a group startup mechanism in that the first member has a direction in which to send its traffic which is correct even if another member has joined the group almost simultaneously. Secondly, it provides a destination for all routers, on-tree or not, to route any multicast packets. Thirdly, once the tree has been formed, the failure of the core is treated in the same fashion as the failure of any other on-tree router. If a hierarchy of routers is defined *a priori*, the tree automatically reconfigures with the secondary router as the core.

The shortcomings of this approach are that the mapping of core addresses to group addresses is not addressed. Particularly, if group addresses are to be assigned dynamically when they become a scarce commodity, it will not sit well with the fact

that the cores are determined statically (administratively). There is no easy way to locate the core. If a multicore tree is envisioned, this problem becomes more acute. Since CBT is proposed as an interdomain routing technique, the interdomain links will quickly become bottleneck links. The same problem occurs when multiple senders send simultaneously.

B. Protocol Independent Multicast

Protocol Independent Multicast (PIM), formerly known as Explicit Source List (ESL) [Ref. 13] is a proposal for dealing with "skinny trees". This refers to a group that is sparsely spread throughout the internetwork. Current schemes for multicasting have routers assume that everyone wants the multicast unless the router determines, via IGMP, that no members exist down a given path. Messages were constrained in their distance travelled by their Time To Live (TTL) parameter in the packet header. This leads to a problem when there are members sparsely populated throughout different networks. To include the distant members, the TTL needs to be increased. This, in turn, requires many more routers to search for paths with no members and so greatly increases the amount of traffic. PIM is proposed to eliminate this inefficiency.

PIM is centered around router(s) designated as Rendezvous Points (RP). For each group a number of routers will be appointed as RPs. A RP is a focal point for members of a group in a given area. Once a receiver is connected to the RP, it can learn who the sources are via an IGMP-Register message. Once this is known, the receiver can elect to receive messages via the shortest path from the source rather than via the RP. Routers are then informed of this decision, so that they can reconfigure if necessary, by an IGMP-ESL message. This technique then provides a means by which a source specific shortest path tree can be formed without flooding the network

as routers find branches with non-members. The authors point out that if there is a large number of sources with low data rates that the group would continue to use the RPs and hence would have a shared tree.

Shortcomings of this proposal are that the routing is determined by Reverse Path Forwarding. This means that the trees created are from the shortest paths from the receivers to the source rather than source to the receivers. This can create incorrect trees when routes are asymmetric. Another limitation is the need for PIM routers. These routers need to become aware of the RPs for each group. If the PIM router does not recognize the group, i.e. it does not have a RP mapping for that group, it assumes that the multicast is not to be supported by PIM. If the TTL is large then the network will be flooded until it learns of the tree.

PIM is designed to help prevent congestion at choke points which would be the root of a shared tree. But PIM requires that all senders include all RPs on their individual tree so that newly joining members can begin to receive traffic. This means that all RPs will have a large amount of incoming traffic whether or not the RP has any members in which to route the traffic. For the input case for the RPs then, it will be no different than a root of a single shared tree except it occurs for every RP that exists for the group. For robustness sake, multiple RPs are required for each group. It appears that congestion will not be relieved at all and possibly traffic is routed where it is not needed.

C. Distance Vector Multicast Routing Protocol

Distance Vector Multicast Routing Protocol (DVMRP) [Ref. 4] is a multicast protocol derived from Routing Information Protocol (RIP) [Ref. 14] and Internet Group Management Protocol (IGMP) [Ref. 15]. It makes use of Truncated Reverse Path Broadcasting (TRPB) [Ref. 1] which is essentially an efficient form of flooding

while incorporating a technique known as "pruning". Pruning prevents the message from being delivered to nets that have no members or are not on the shortest path from the sender to other routers.

TRPB makes use of distance vector routing tables to minimize duplicate packets sent out on the network. These allow "parent" and "child" relationships to be set up. A child router depends on the parent router to forward packets from a given sender, i.e. the parent router is on the shortest path from the child to the sender. Neighboring routers share their information on distance to the sender so that they can discover the shortest path to the sender and also determine which is the dominant router if they both service the same net. The router that is determined to be the dominant router is the one to service that net. The other router will not send any packets to that net as they would only be duplicates. If both routers are equal distant to the sender then the dominant router is the one with the largest IP address.

In addition to parent and child relationships between routers, there is the possibility of a "leaf" relationship between a router and a net. A net is considered a leaf if no other routers consider that net as part of the shortest path to the sender i.e. there are no parent/child relationships across that net. If a network is determined to be a leaf and there are no members of the group on that net, then the router correctly determines that there is no need to deliver the multicast packet to that net. This again helps reduce the number of needless packets delivered.

DVMRP is for use only within an autonomous system. As can be seen every router receives the packet. TRPB merely prevents the delivery of duplicate packets. The only areas a packet is not delivered to is a leaf network with no members, which gets pruned. Future implementations of DVMRP hope to use Reverse Path Multicasting (RPM) [Ref. 1]. This is more efficient because it will enable the pruning of routers that have no members of the group downstream of itself.

Disadvantages of DVMRP are that the packet may be delivered to locations where it is not needed. It is limited in scope to within an autonomous system. Current methods use distance to the sender, for asymmetric links, the shortest path may not be discovered. It has been mentioned that it is possible to use the reverse distant cost so that the actual forwarding cost from the sender would be used and avoid the asymmetry problem. But this will double the size of routing tables because unicast uses the previous metric to determine the shortest path to the receiver.

D. MOSPF

MOSPF [Ref. 3] is the multicast extension of the link-state protocol known as OSPF [Ref. 16]. This method is for use within an Autonomous System (AS) and not over the entire Internet. With link-state, all routers know the distance to all destinations within the AS. To add a multicast capability, a new link advertisement message is made. This message lets all the other routers know where members of a multicast group lie. MOSPF utilizes IGMP for the routers to discover hosts belonging to the group. This means that every router can determine the source based multicast tree for every source/group pair in the AS. Once this is done the router caches the output interfaces for each source/group pair for future use.

To add the ability for multicasts to enter or leave the MOSPF AS, the border routers, (those adjacent to neighboring AS's become wildcard members, that means that they are automatically members of every group. This allows all multicast messages to reach the next domain. The TTL of the message can be set so as to limit the scope of the message.

The shortcomings of this approach are the extensive calculations required for every router to determine the source based tree. This will be done for every source for the group. The author points out that these calculations will be made on demand

TABLE 3.1: EXISTING APPROACHES VS. DESIGN GOALS

Goal	CBT	PIM	DVMRP	MOSPF
Utilize <i>a priori</i> information	N	N	N	N
Automatic locating of the DC's	N	N	Y	Y
Flexible choice of tree type	N	Y	N	Y
Tree setup with QoS	N	N	N	N
Support multiple routing protocols	Y	Y	N	N
Minimal tree state information	Y	N	Y	N
Minimal router processing	N	Y	N	Y

so as to spread the computations over time and prevent unnecessary work for the processor in figuring out trees for non-existent source/group pairs. In addition when the state changes in the net, it is not clear if the entire tree will need to be recalculated for every source or if the computations can be made so as to only update the affected interfaces of the source/group pairs. Membership reports are sent to all routers so unneeded information may be promulgated and take up memory.

Another point is with the use of source based trees. The caches of the routers will need to be large because of the possibility of groups with a large number of sources. If more than one source for a given group uses the exact same interfaces at a router, there will still be separate cache entries for each source.

E. Comparison against Design Goals

See Table 3.1 for a comparison of the above mentioned approaches to the design goals. DVMRP and MOSPF are considered to automatically locate the Distribution Centers because they automatically use a SST.

IV. General Approach

In view of the shortcomings of current approaches described in the previous chapter and the design goals outlined earlier, the proposed approach has been developed with distinct aspects. They are: actions taken by the network based on *a priori* information available about the upcoming interaction, group-specific location of one or more cores and establishment of corresponding trees, changes in the trees' topologies due to the dynamic multicast group membership, and reservation of resources. In this chapter, we describe each of these aspects and relate it to the design goals justified earlier. The mechanisms to be provided by the network to implement this approach are described in the next chapter.

A. Assumptions

Following are the assumptions in our approach. We have assumed that the network links are bidirectional and the costs are symmetric. In other words, the cost of going from A to B is the same as the cost of going from B to A. While it is claimed [Ref. 13] that most of the current Internet links are symmetric in their capacities, it is unrealistic to expect that the traffic load on any link be symmetric. This makes our assumption rather severe; however, we plan to address this problem in the future. We do outline the impact of relaxing this assumption in the last chapter.

In the current Internet multicast, membership in a group *only* affects a member's ability to receive multicast traffic to the group. However, members as well as non-members *can* send to a group. We assume the existence of some security mechanism that forces a sender to first become a member.

We also assume that if two senders are topologically close, the local routing algorithm will *not* give two *almost independent* paths to a distant node (such as a core). The case when this happens is illustrated in Fig. 4.1(a). This permits us to not worry about the case that two nearby sources will lead to reservations along separate paths. In case of CBTs (or shared trees), this assumption merely implies that, as multicast packets progress towards the core, they seek the shortest path (SP) as well. One problem with attempting to distinguish the SP to the core from the SP to the group (which may be shorter as illustrated in Fig. 4.1(b)) is that it is difficult to determine at what point the SP to the group is to be preferred over the SP to the core. Determining this cross-over point requires the more general trade-off between resource consumption and delay in the tree construction mechanism itself. We have elected to address this trade-off at the group level rather than individual member level by locating multiple distribution centers for a group if necessary. If a single distribution center exists for a multicast group as in CBT [Ref. 2], the difference between the distances to the nearest group-aware router and to the core is likely to be high. In this case, SP to the group is likely to result in reservation of fewer resources on the whole whereas SP to the core will minimize the average delay.

With a dynamic group membership, it is difficult to make a packet seek the shortest path to the group in practice (it is not known which way leads to the closest member). A group member that is not directly connected to any router that is aware of the required group, relies on some protocol like IGMP [Ref. 15] to reach a router that is *group-aware*. In the future, we expect that the new member could rely on some hierarchical global group membership service that associates a group name with the address of the nearest distribution center.

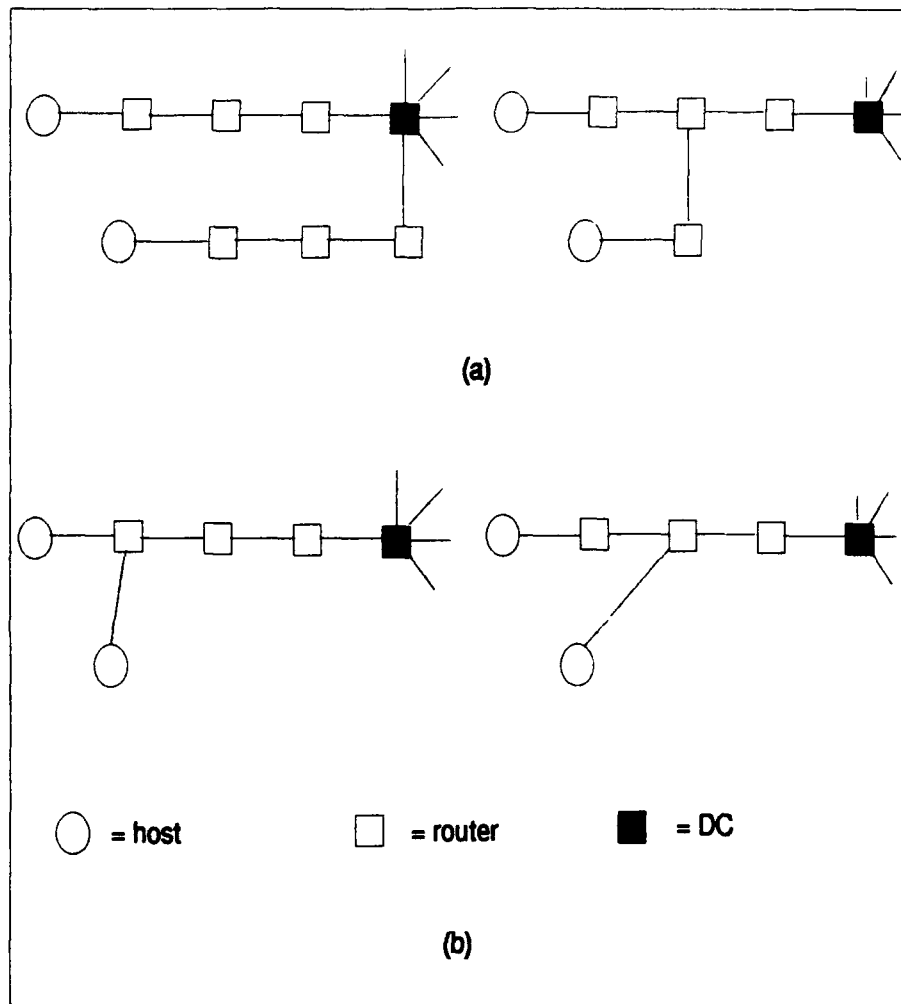


Figure 4.1: Illustration of assumptions

B. Use of *a priori* Information About the Participants

In our approach, depending upon the nature of the multiparty interaction, it is desired to minimize network resources that need to be reserved by striking a compromise between shared CSTs and SSTs. The nature of a multiparty interaction is determined by the following three parameters: reservation requirements of individual CPs, number of simultaneous senders, and the geographical distribution of members. We anticipate a network entity, called a *scheduling register* (SR), similar to the session directory (sd) tool developed by Van Jacobson [Ref. 17], that permits participants to register and declare these requirements. Every participant should know its reservation requirements (bandwidth) and its address (location). It can also be expected to know its sending requirements as they relate to the role (function) it will play in the interaction. This will permit the SR to determine if this sender is required to send concurrently with the other senders.

Based on this information supplied by all the participants, the SR can deduce the amount of bandwidth required by each and the distribution of the CPs. The SR can also group all the CPs into subsets such that no two CPs in a subset send at the same time and, at any time, there is at least one CP from the subset with send traffic. We refer to each of these groups as a *critical set of participants* (CSP).

This grouping is justified by our viewpoint that the CPs of any interaction group can be grouped into one or more CSPs depending upon their sending and reservation requirements, and geographical distribution. If a certain CP is expected to be sending throughout the interaction, a sender-based SST should be formed to deliver its traffic most efficiently (with the least delay and minimum reservation of resources) to all the receivers. On the other hand, if a subset of CPs is likely to have only one sender among them at any time and/or send for only a part of the interaction, they should share a tree. Moreover, CPs that are in distant and separate routing domains

should not share a tree. Thus, our approach proposes a way of creating an efficient combination of shared and sender-based trees for a given interaction.

C. Locating a Set of Distribution Centers

The shared tree approach is criticized for being prone to congestion on the backbone links in case of multiple simultaneous interactions. For rich topologies, Wei and Estrin [Ref. 5] show that sender-based SPTs distribute bandwidth requirements more evenly than CBTs although their overall requirement for a group is higher. This is due to the fact that the core for the shared tree is located administratively regardless of the locations of the members.

It has also been shown that constructing a multicast tree naively is almost as good as some optimized technique [Ref. 18]. However, this analysis does not account for other simultaneous multicasts, and therefore, cannot be applied to multiple simultaneous multiparty interactions. Wei and Estrin [Ref. 5] claim that a member-based SPT is almost as good as an optimal CBT. However, which member is selected is not specified. It seems that for widely distributed groups, this member's location will be critical to the tree quality.

In our approach, we specify a mechanism for the network to determine a set of router locations as distribution centers for the interaction. It provides a way of selecting a distribution center (which performs the same function as the core of a CBT or a rendezvous point of PIM) for each CSP. Being dependent on the CP locations in a CSP, such selection of the center leads to the construction of the shared CST in a manner that distributes the resource usage across the network uniformly. This can be achieved by maximizing the resources that remain available in the network while selecting the center location. Our approach of dividing the CPs into CSPs for formation of a shared trees based on the CP attributes includes cases when a CSP

must contain only one CP and the core gets located at the sender itself resulting in a sender-based tree. Essentially, this leads to the establishment of multiple centers with a QoS tree originating at each. We describe the approach to selecting the location of a center for a single CSP below.

1. Selecting the Location of a Distribution Center

The principal requirements for the location of a distribution center are that it should be fast, scalable, and produce trees with the required quality while using as few network resources as possible. It should also produce trees that distribute center locations in the network based on the locations of the CPs. In our approach, we rely on an administrative mechanism of SR to inform the CPs about the CSP they are in. The SR also provides a CP id to each CP within a CSP. Each CP identifies itself as belonging to a CSP id assigned by the SR. The routers corresponding to CPs with the same CSP id then enter a distributed pair-wise selection process that results in the selection of a center. For each CSP, the SR describes the complete hierarchy of pairings used for center selection. Using this hierarchy, a single center is located and a shortest path tree is established as the tree shared by the senders in that CSP of the multicast group.

There are several aspects of such center selection that need to be specified. Firstly, the SR has the responsibility to provide the complete pairing hierarchy for all the phases of the selection process. Each phase represents a pair-wise selection of a router location for a pair of router locations. The router location selected (called the winner) for a pair can be some router (preferably at the center) along the path between the two. Since the home routers of the CPs themselves participate in the first phase, such pairing can be done based on the inter-CP distances by the SR. Note that this is possible *only* in the first phase since CP locations are known to the SR.

However, the locations selected in the second phase onwards are not known *a priori* to the SR. Therefore, pairing in the subsequent phases is done without any additional distance information. A deterministic algorithm, based on the CP id within a CSP, determines the pairs in each of the phases. The SR executes this algorithm and distributes the selection hierarchy to each CP in a CSP. The detailed algorithm is given in the next chapter.

In the first phase, locations that are farthest apart are paired off. The SR can derive such information easily from the unicast routing databases at a router. The justification for pairing off the most distant locations is that the central location is determined more quickly. Using the same argument, we require that, if a winner has no partner in a particular phase (there are an odd number of them), it must be paired off at the earliest opportunity in the subsequent phases. This ensures that a far-flung location will have the least impact on the final winner's location. Note that if such a location is paired-off when a center is determined for all the other locations, the final center location can be displaced considerably from the network center of the CSP as compared to the case when it is paired-off at the earliest opportunity. Generalizing this, we require that the SR determines an inverted tree of locations, with the CPs at the top (leaves), that minimizes the number of successive *bye* phases for a particular location.

Once the centers are selected for each CSP, each winner informs the SR of the group id for the interaction and the CSP id for which it won. The SR maintains a map of the group id, CSP ids and their associated centers. When all the winners have reported to the SR, the SR informs all the CPs of the complete list of center addresses for all the CSPs of the group using unicast. Center selection is illustrated in the example that follows.

2. Illustration of the Selection of a Center

Figure 4.2 shows a hypothetical network of 12 nodes with 5 critical participants. The CPs are indicated by filled squares and the network routers are represented by hollow circles. The link costs are given alongside the edges.

Assuming all the CPs form a single CSP, the SR determines the selection tree as given in Figure 4.3. Note that none of the routers in any phase receives a bye in more than a single consecutive round. We expect that the pairing in the second phase onwards is not as critical to the final location as ensuring that there are no consecutive byes. The initial pairing is expected to place all the second phase pairs in the same vicinity if the CPs are evenly distributed in the domain. In case of non-uniform clustered distributions, this approach naturally makes the core gravitate towards the heavier cluster.

3. Functions of the SR

Based on the above two aspects of the approach, the functions of the SR, related to the setup of an interaction, can be listed as follows:

- Accept registration from participants up to a certain time before the scheduled start of an interaction depending upon the cost to the user. If strict QoS guarantees are required when the network load is high, the SR closes registration and begins scheduling earlier so that resources can be blocked earlier. Such users can be charged more depending on how long the resources are reserved.
- Determine the CSPs from the CPs based on their reservation requirements, simultaneity of sending, and the administrative domains in which their locations reside. Assign a CSP id to each CP and a CP id within each CSP.

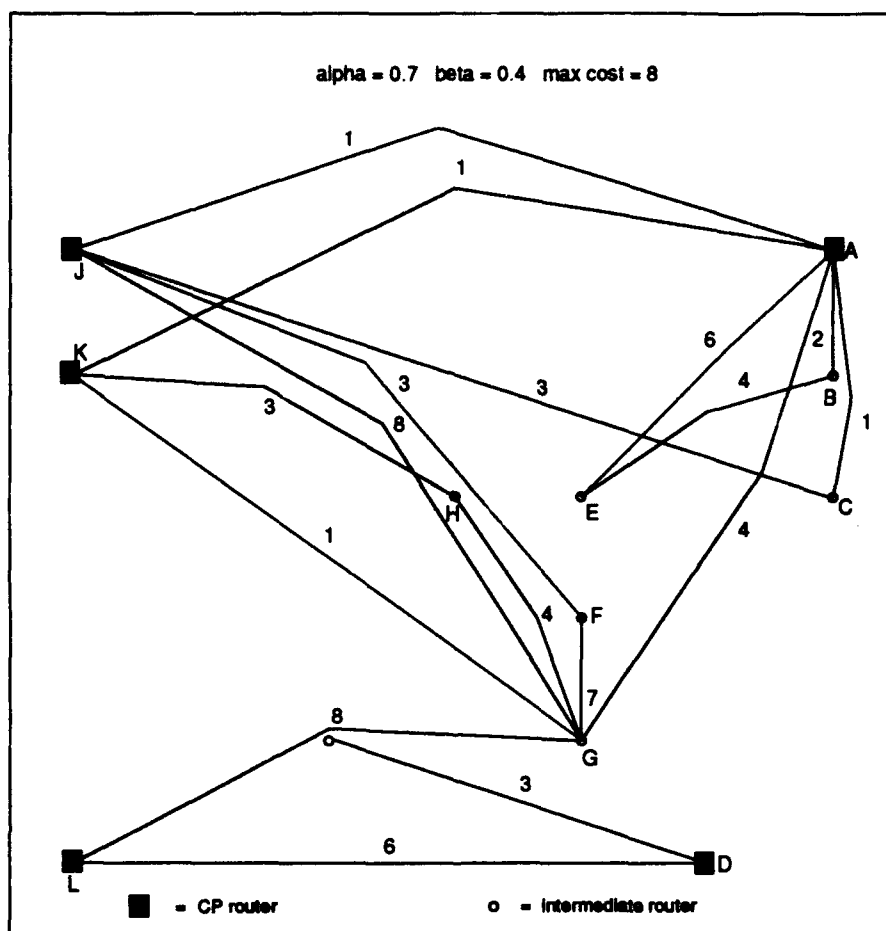


Figure 4.2: An Example Network

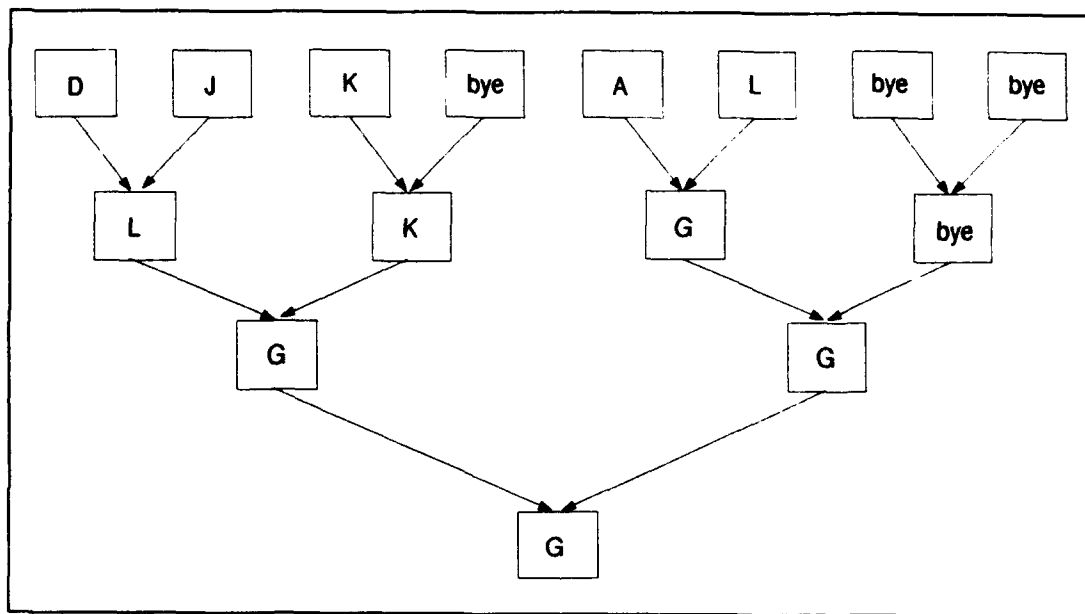


Figure 4.3: Selection Tree for the Example Network

- Determine a selection hierarchy for pairing of CPs within each CSP and propagate it to each member of the CSP.
- Receive the centers' addresses for each CSP at the end of the center selection for each CSP. Maintain a mapping of the group id and the CSP ids along with the associated centers. Propagate this mapping to each CP of the interaction.

D. Tree Construction

After the centers are located, the tree establishment process and making all the CPs, as well as the non-CPs, aware of the local core proceeds as follows.

All the CPs join all the centers explicitly for receive traffic. This joining can proceed in a manner similar to the CBT approach [Ref. 2]. Thus, a center-specific shortest path tree gets formed for every center selected for each CSP. If there is only one sender in a CSP, this tree will be the same as an SST for that sender.

Based on the CSP id assigned to each CP by the SR initially, a CP selects the center address corresponding to this CSP id from the list supplied by the SR as its home center. It directs all the send traffic towards this center along the the corresponding CST. Since it is the responsibility of the other CPs to join this center for receive traffic, the sender need not be concerned with being able to send to the other CPs. All CPs that belong to some other CSP id attach themselves to this CST during their join processing. It is expected that, by having each receiver join each center separately for receive traffic, fairly dissimilar paths would carry the transmissions originating at CPs in different CSPs to a given end-receiver. This will alleviate the congestion. Essentially, by limiting an appropriate number of potentially concurrent senders to a CST, this approach helps reduce the potential for congestion while permitting the efficiency resulting from the use of shared trees.

Dynamic membership management is similar to that in the CBT approach.

Every router is regarded as either *on-tree* or *off-tree* with respect to a specific CST. An *on-tree* router knows which of its outgoing interface leads towards the center of the CST. Thus, it is likely that a router is *on-tree* for one CST of an interaction, but *off-tree* for another CST of the same interaction. A router that is *on-tree* for *any* CST of an interaction, is called a *group-aware* router. Such a router maintains state for a group by maintaining a list of all the centers for the group-id. Since the number of CSPs is expected to be small, this state does not represent a scalability problem.

The proposed approach is similar to PIM [Ref. 13] in that a list of locations is maintained. The important difference is that, in PIM a list of all the senders is maintained at the RPs and a new sender must register with every RP. Clearly, this requires greater latency in joining a group and larger state to be maintained in each RP than the proposed approach. Our approach requires a new sender to attach to one of the centers already known to all the *group-aware* routers. If the new sender is likely to require its own SST because it is going to be sending all the time, it has one of two alternatives. Firstly, it can register itself with the SR, become a part of all the group aware routers' state, and acquire a tree with the required QoS. Alternatively, it can attach to one of the existing centers causing temporary congestion, and get the reservations on this tree upgraded eventually. In the second option, the join latency is less and the state maintained by all the routers does not grow with the number of new senders requiring an SST. A new sender is not guaranteed of a tree with the required QoS anyway.

However, this does not address the issue of changing the number or locations of the distribution centers once they have been determined at the start time. In the proposed approach, no provision is made for reconfiguring the distribution centers. It is assumed that the number of unanticipated senders of an interaction does not change excessively during the life of an interaction that the distribution centers selected by

the SR are no longer applicable. We regard this as an area for further investigation.

A new participant sends its join request to its directly connected router which follows IGMP to propagate it to its own directly connected routers. This presents the possibility of burdening routers that have nothing to do with a group and are unlikely to lead to any *group-aware* router by making them propagate this join request. This makes any IGMP-based solution inherently non-scalable. Ideally, there should exist a hierarchical name service for multicast group names that any router has access to. It could then query this name service to discover the group-name to list-of-centers mapping for the group it wants to join. Such a service, although envisioned [Ref. 19], does not exist today. In any case, the first *group-aware* router to receive the join request responds with a list of the distribution centers for that group. A new receiver then joins every center explicitly and a new sender selects a home center and joins the corresponding center-specific tree explicitly. This joining could progress in the same manner as the CBT join processing.

E. Reserved Routing

The ability to guarantee a quality of service depends on resource reservation. This leads to two possible methods in forming a multicast tree. The reservations can force the tree or the tree can force the reservations. In the first case, a branch of the tree would not be established unless sufficient reservations can be made along that link. This method compels an all or nothing approach. A connection is only created if the quality of service desired is attainable at that time. This technique would create a greater establishment delay. The other method is to let the routing protocol create the multicast tree and then attempt to make reservations on the links already formed. The drawback here is that the route chosen may have insufficient reservable resources and hence may inhibit the desired quality of service.

Both methods could allow a temporary degraded level of service and then attempt to find additional resources. This may be done by acquiring resources freed up on the existing branch or finding an alternate path that provides a better level of service. Searching for the path with the best reservation level may be time consuming. Therefore it is deemed better to create the tree first and then obtain the resources. This may result in a degraded service but minimizes establishment time.

Since our approach uses *a priori* information, the task of acquiring reserved resources has the following three phases:

CST construction based on resource availability:

As per the design goal of integrating reservations with tree construction, each CST gets constructed based on the resource availability. During the selection of the center location, the designated member of a pair sends a probe to the other member. This probe message is sent with the *type-of-service* (ToS) option and parameters set as per the local routing protocol [Ref. 3]. This requires the probe to seek the path with the shortest path according to the ToS parameters. If the ToS parameter used is the *unreserved bandwidth*, the winner will get located along the path that has the most unused bandwidth available. Use of ToS-based routing in this stage will make the location selection mechanism sensitive to the current network load distribution.

Bandwidth reservation on the CST formed for each CSP:

Once the center is selected, both the senders as well as receivers join each CST explicitly. The responsibility for establishing reservations from the senders to the center is borne by the senders and between the receivers and the center is

borne by the receivers. This permits sharing of the reservation responsibility instead of a purely receiver-initiated approach as in [Ref. 6]. With this approach, the reservations made by the senders are held for the duration of the interaction.

The senders acquire reservation on the CST as follows: each sender initiates a *count* message that is *concast* along the CST. As the count of this *count* message gets accumulated upstream along the CST, each *on-tree* router initiates the required reservations on its outgoing interfaces. Reservations on the incoming interfaces of a router are dictated by the sending end of the interface. Here, we assume that directly connected routers are able to execute some low-level point-to-point protocol that permits the reservations on the incoming interface be dictated by the sending end.

A receiver sets up a path with reserved resources from the center as follows: each receiver sends a join message unicast to the center. The first *on-tree* router encountered sends a join acknowledge (as in CBT) along the shortest path based on a ToS-related parameter, such as the available bandwidth (unlike in CBT). This message establishes the maximum between the required and the available bandwidth on each interface it passes out of.

Acquire and release resources for incoming and outgoing participants:

The reservations required by participants that arrive after the start of the interaction are acquired in much the same manner as the receivers and senders that are CPs. The unplanned senders reserve a path to their home center and the unplanned receivers reserve a path from each center via the join ack resulting from their join messages.

Shortest paths computed based on the ToS-related parameters ensure that the CST topology remains sensitive to the changing network load. It must, however, be ensured that the first multicast data packet traversing a new reserved path is preceded by a control packet that acquires the reservation.

V. Protocols for Locating the Distribution Centers

Several protocols are required to support the multicast tree construction described in the previous chapter. In this chapter, we describe the protocol to locate the distribution center for a CSP in detail while qualitatively describing the others. First of all, we list all the protocols required.

1. A protocol is required for all the potential critical participants to register their attributes with the special application SR. The information exchanged between the participant and the SR must include an identification of the interaction, the location of the participant, its role in the interaction, and other attributes such as the QoS parameters desired. This protocol must also address how the participant will receive the selection hierarchy and the list of centers once they have been selected. We call this protocol the *participant registration protocol*.
2. A protocol is required for the registered participants in a CSP to participate in the pairwise center selection process. This protocol must address how the winners in a phase receive location information about their partners. We call this the *center selection protocol*.
3. A protocol is required for each sender CP to attach itself to its home center and for each receiver CP to join the CSTs for all centers. It must also address how the dynamically arriving senders and receivers receive the center list and how the senders choose their home center. We call this the *tree construction protocol*.

4. A protocol is required to reserve the required resources on the tree branches for each CST. This protocol must address how each router, depending on its location in the tree, determines the bandwidth to be reserved on each outgoing interface, how directly connected routers negotiate for bandwidth reservation, how the *type-of-service* parameters are treated by each router, and how the reservation requirements are met as resources become available. We call this the reservation protocol.

While we have qualitatively described our approach to the services to be provided by each of these protocols in the previous chapter, we limit ourselves to the detailed description of the first two in this thesis. The tree construction protocol is expected to build upon the ideas presented for CBT [Ref. 2] and PIM [Ref. 13]. The reservation protocol is expected to build upon the ideas and approaches presented in the literature on the QoS related reservation protocols summarized in Chapter VII.

A. Participant Registration Protocol

This protocol supports an administrative mechanism. As mentioned previously, we anticipate the existence of a internet-wide special application, called the *scheduling register*, similar to the session directory tool [Ref. 17]. Interactions as well as participants along with their QoS attributes and their role in the interaction are posted to the SR ahead of the start time of the interaction.

The SR primarily deals with participants that have planned their participation in an interaction prior to its start. It does not deal with participants that arrive after the interaction has started. Registration for an interaction is closed at a certain time prior to the start, as determined by the network administration. Based on the QoS requirements and other attributes for the *registered* participants of an interaction, the SR determines if a participant is critical or not. Once all the CPs have been

determined, it groups them into subsets, referred to as critical set of participants (CSP), and assigns a CSP-id to it. This grouping occurs according to the maximum number of concurrent senders permitted in a CSP for the interaction, the geographical distribution of the CPs, and their reservation requirements. The SR is generally aware of the internet topology and its administrative boundaries and is capable of mapping host names of the CPs to networks. This knowledge is used for the grouping of CPs into CSPs each of which spans a single administrative domain. The CSP-id that a CP is grouped into is sent to the CP, along with all the other participants in the CP, using a unicast. The exact algorithm used by the SR for determining the CSPs is to be addressed in the future.

Once the CSPs are determined, the SR performs the task of determining the selection hierarchy within each CSP for locating a center for that CSP. As outlined earlier, the selection of a center proceeds by a pairwise selection process. In order that the message exchange required to carry out this selection is minimized, the SR provides the complete selection hierarchy as illustrated in the example of the previous chapter.

The SR uses its knowledge of distances and domain topology to form the initial pairs of CPs. The initial pairing could be based on the CPs that are nearby or that are far apart. Pairing of a CP with the farthest CP appears beneficial because, in case the CPs are situated symmetrically, the center may be found very quickly. Also, pulling in CPs that are far away from a cluster in the initial phases will make the center gravitate toward the cluster. The pairs in the subsequent phases are determined according to the deterministic algorithm outlined in Fig. 5.1. It is to be noted that, due to the initial placement of *bye* positions, the number of successive *byes* a winner gets is minimized. The algorithm SR uses for determining who gets a bye in the first phase is given in Fig. 5.2. A *bye* position propagates in the hierarchy

```

SelectionHierarchy for CSP id = cspid at SR
  numcps = number of CPs in cspid;
  number of phases  $n = \lceil \log_2 \text{numcps} \rceil$ ;
  /* slots[i][j] is the jth winner in phase i; */
  initialize slots[0][j]  $\forall j \in [1, 2^n]$  with bye
    using ByeDetermination;
  initialize pairs of empty slots[0][j] with CP pairs
    in decreasing order of distance;
  for  $i = 1$  to  $n$ 
    number slots in phase i, numslots =  $2^{n-i}$ ;
    for  $j = 1$  to numslots
      slot[i][j] = winner of slot[i - 1][2j - 1] and slot[i - 1][2j];
  end SelectionHierarchy.

```

Figure 5.1: Algorithm for Determining the Selection Hierarchy

until it meets a winner from the previous phase. When a pair has one *bye* position, the winner from the previous phase naturally enters the next phase as the winner.

Depending upon the cost of the multiparty interaction to the customer, the the SR determines the instant prior to the start of the interaction for propagating the selection hierarchy to all the CPs in a particular CSP. This propagation is performed using unicast messages. The use of point-to-point messages does not represent a drawback since the number of CPs is expected to be small. Also, this occurs only at the start of the interaction. Once the CPs receive this pairing, the center selection mechanism gets triggered.

B. Center Selection Protocol

This is a protocol for locating a set of centers, one per CSP, in the network for an interaction automatically, rather than administratively. The protocol essentially selects one out of a pair of routers until a single router is selected as the location of the distribution center. For each center, a shortest path tree based on unicast shortest

```

ByeDetermination by SR
  number of phases  $n = \lceil \log_2 \text{numcps} \rceil$ ;
  byecount =  $2^n - \text{numcps}$ ;
  count = 0;
  while byecount > 0
     $m = 2^{\lceil \log_2 (\text{byecount}) \rceil}$ ;
    for  $i = 1$  to  $m$ ;
      set slots[0][ $2^{n-\text{count}} - m + i$ ] as a bye position;
    byecount = byecount -  $m$ ;
    count = count + 1;
  end ByeDetermination.

```

Figure 5.2: Determining the Bye Positions in the First Phase

path computed according to the *type-of-service*-related cost gets constructed using the approach described in the previous chapter. If there is only one sender in the CSP, this protocol selects the center at the same location as the sender. Thus, it yields a source-specific tree. If there are multiple senders, the center-specific tree is shared by them. Several aspects of this selection protocol are described below.

We have noted that the SR sends each CP in a CSP the group id, CSP id, id of the CP within the CSP, the number of CPs in the CSP, their locations, and the complete selection hierarchy. By the complete hierarchy, we imply that each CP *knows* its partner in the first phase and can also determine which pairs' winner its own winner will pair up with in each of the subsequent phases. At the end of the selection of winners for each phase, some mechanism must be present so that each winner comes to know of the location of the other winner it is to pair up with. Of course, a winner can determine if it is the eventual winner for the CSP based on the number of phases that have completed since this number is known deterministically.

There are two alternatives for determining the location of the partner. Among all the CPs that the winner of a phase represents, one CP can be designated as

the point of contact (poc) deterministically, say, the lowest CP id in that CSP. The winner can send its location information to this poc from where the other winner collects it. Although, this enables the winners in each phase go to known locations, it is not desirable since the poc could fail. We choose the following approach.

The other alternative is to ensure that, at the end of a phase, every winner knows all the other winners. Thus, every winner announces its location to the leader in each pair of the previous phase. The leader of a pair can be designated as the CP with lower CP id. Due to the deterministic selection hierarchy, a CP can determine, purely based on its id and the number of CPs, whether it is the leader of a pair. It sends a probe message to its partner who echoes it. As this message makes its way back to the sender, the route gets recorded.¹ The leader determines the winning router for the pair from the recorded route and informs all leaders of the other pairs of that phase of its winner's location. Since every leader performs the same actions, all the leaders arrive at the complete list of winners from that phase. Each leader then distributes the following items of information to the winner from its pair: the group id, the CSP id, the phase number, its id in that phase, and the locations of winners in the phase completed. When the winner receives this information, it determines its partner and starts the next phase by communicating with its partner.

The winner of the final phase communicates with the SR its selection as the center and the CSP id it represents. When winners from all the CSPs communicate their selection to the SR, it distributes the group id and its associated list of centers to all the CPs. The center-specific trees then get constructed according to the tree construction described in the previous chapter.

The number of phases required for center location protocol described here to select a center is proportional to the logarithm of the number of CPs in a CSP. This

¹This is where the assumption about the symmetry of the path becomes crucial.

makes the selection mechanism fast and scalable in terms of the number of CPs. Use of the *type-of-service* field also makes this protocol sensitive to the current load conditions in the network.

In the next chapter, this approach is used for selecting centers for large hypothetical topologies. The results show that the average delay seen by the participants using CSTs increases only marginally as compared to the SSTs (which yield the lowest delay) while fewer resources are consumed.

VI. Performance Evaluation

Three performance parameters have been used. *viz.*, delay, cost, and traffic concentration. How appropriate are these given that the future internetwork is most likely to have a rich topology, high bandwidth, but with a high probability of congestion. Are these parameters appropriate when the primary yardsticks are how long it takes to establish a connection and how many are maintained simultaneously?

We mainly evaluate the performance of our core location algorithm. The algorithm is applied to a set of random graphs.

A. Generation of the Network Topology

A random network is created by the use of Waxman's RG2 algorithm [Ref. 20]. First a maximum value for the cost between any two nodes (L) and the number of nodes (N) is established. Next every pair of nodes (i, j) is assigned an integer cost ($d_{i,j}$) utilizing a uniform distribution from 1 to L . The probability (p_{ij}) that a link exists between a node pair (i, j) is determined by $p_{ij} = \beta \exp^{d_{i,j}/(\alpha L)}$. If the link exists then its cost is $d_{i,j}$. The parameters, α and β , are defined in the interval (0,1]. A small value of α will cause a relatively greater number of low cost links.

The node degree, (λ_i), (the number of links from a given node) for node i is approximated by

$$\lambda_i \approx \sum_{j=1, j \neq i}^N p_{ij} = \beta \sum_{j=1, j \neq i}^N \exp^{\frac{-d_{i,j}}{\alpha L}}$$

Since $d_{i,j}$ is uniformly distributed integers between $[1, L]$ there will be approximately $(N - 1)/L$ of each possible value of $d_{i,j}$. This allows the following revision.

$$\lambda_i \approx \frac{\beta(N - 1)}{L} \sum_{k=1}^L \exp^{\frac{-k}{\alpha L}}$$

Letting $\exp^{-1/(\alpha L)} = \rho$ and observing that the above equation is a finite geometric sum of ρ yields

$$\lambda_i \approx \frac{\beta(N-1)\rho(1-\rho^L)}{L(1-\rho)}$$

Since λ_i can be approximated by this method for every i , the average node degree λ_{avg} for the entire graph can be approximated by this formula.

$$\lambda_{avg} \approx \frac{\beta(N-1)\rho(1-\rho^L)}{L(1-\rho)}$$

Solving for β results in the following

$$\beta \approx \frac{\lambda_{avg}L(1-\rho)}{(N-1)\rho(1-\rho^L)}$$

If further simplification is desired ρ^L can be approximated by 0 and $N-1$ can be approximated by N for $L \gg 1$ and $N \gg 1$ respectively.

This solution worked very well but a study on the variation was not performed. When average node degrees ranging from [3,5] were used, the graph's λ_{avg} was in the desired vicinity but there would still be some nodes not connected to all other nodes. These nodes were very few in number so the graphs were used provided all of the CP's were connected. Doing this effectively lowered N by a small amount. This would not adversely affect results since the results are not specifically tied to the number of nodes.

To build the steiner tree the two closest participants were joined together by their shortest path link. Next the participant closest to the connected participants or any of the nodes along the path connecting the participants was added to the tree via its shortest path to the nearest node on the tree. This procedure was repeated until all participants were a part of the tree.

B. Simulation Results

Simulations were carried out to determine the drain on network resources for simultaneous senders. For this, 120 nodes were selected with the maximum cost between connected nodes being 8 and α was selected to be 0.125 for all cases. Node degrees were selected to be in the vicinity of 3, 4, and 5 and then β was calculated as shown above for each case. For each node degree, a tree was created that connected 3, 9, and 16 CP's together. The number of simultaneous senders was ranged from 1 to CP -1. Since it is not necessary for a sender to receive himself, there is no difference in network resources utilized for CP -1 or CP simultaneous senders. The tree cost for SST (Source Specific Tree), CST (Center Specific Tree), and ST (Steiner Tree) were calculated for each situation.

Tree costs were calculated by determining the number of sources that used each link. This was done for both directions of each link. Let S_{ij} be the number of sources that use the link from node i to node j and d_{ij} be the cost to use that link. Let ss be the number of simultaneous senders. The tree cost for that link, is then given by $tc_{ij} = \min(ss, S_{ij})d_{ij}$. The total tree cost, tc_{tot} is then $\sum_{i=1}^N \sum_{j=1}^N tc_{ij}$. See Appendix for the code listing.

Figures 6.1 and 6.2 provide representative results. The SST starts out with a higher cost, peaks more rapidly but levels off. Both the CST and the ST tend to increase at a constant rate. When the number of simultaneous senders becomes large it is possible for the cost of the CST to exceed that of the SST. The shape of the SST is due to a greater number of shared links but these links are shared only among a few senders. This differs from the CST and ST whose shared links are used by most senders. These figures show that as far as total network cost for reserved resources, it is better to use a shared tree.

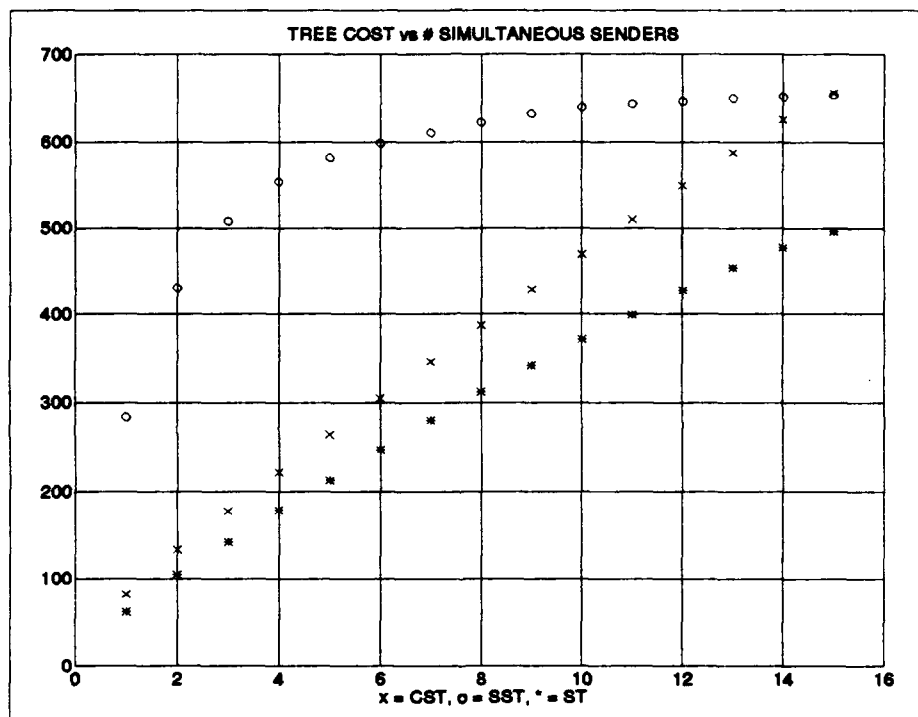


Figure 6.1: Tree cost for $\lambda = 4$ and 16 CP's

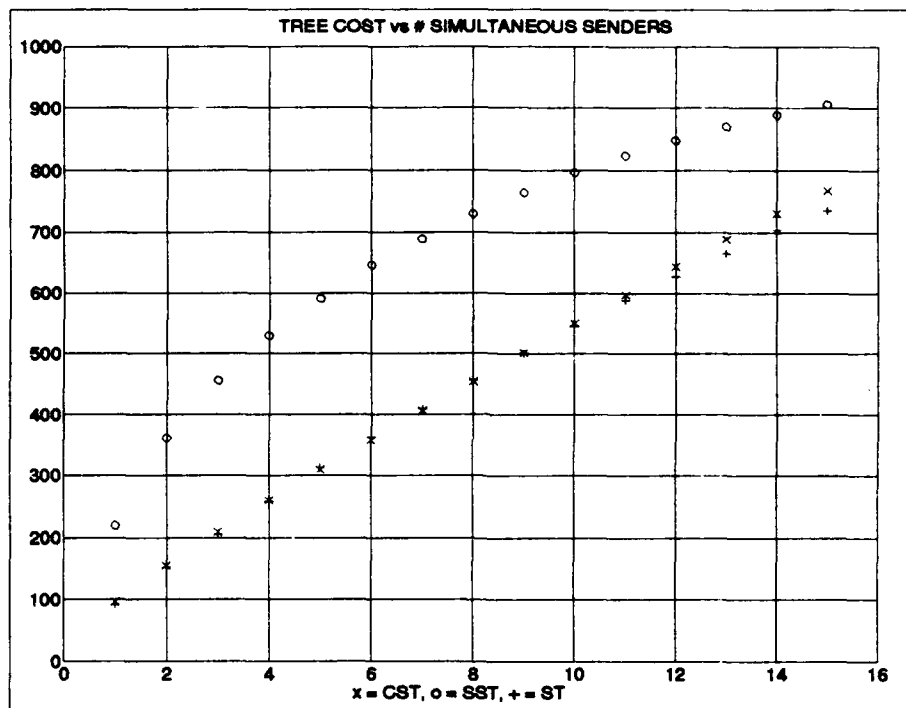


Figure 6.2: Tree cost for $\lambda = 3$ and 16 CP's

C. Qualitative Comparison With PIM

PIM requires that every receiver learn of each sender only through an RP. This implies that an RP, which currently has no receivers that get their multicast traffic through it, must keep listening to every sender. In addition, a sender must keep sending traffic to every RP, even if each one of them does not have any members attached. Let N_s , N_r and N_c be the number of senders, receivers, and RPs (in case of CSTs - number of centers) respectively. In PIM, a new receiver gets all its multicast traffic from a single RP regardless of the number of senders. In the proposed approach, each receiver attaches itself to all the centers. A sender sends its traffic to only one center. Thus, PIM apparently scales well with the number of receivers and our approach scales well with the number of senders. Since there will typically be a greater number of receivers than senders it would be better to scale to senders. A rough estimate of the size of the solution for PIM is $N_s N_c$. This is provided that all receivers elect to receive their traffic through the RP. If all receivers opt to gather their traffic from every source, the metric becomes bounded by $N_s N_r + N_s N_c$, a rather significant increase. For the proposed method it is $N_r N_c$. This does not take in to account the efficiency of multicast. For PIM, there is no multicast efficiency for the first metric given. For the second equation there should be a factor of improvement with multicast. For the proposal there should also be a similar factor of improvement.

VII. Techniques to Guarantee QoS

The following proposals lay the groundwork for the reservation protocol required in this proposal. They all represent techniques to deliver performance bounds on real time traffic.

The first two techniques examined, desire the network to provide some of the buffering enroute to the destination. This spreads the buffer requirements over the entire path travelled by the packet and alleviates the large buffer requirement at the receiver. As a result these methods are also non-work conserving because packets will wait in the buffer until they are released by the protocol, regardless if there is no other traffic being transmitted. The advantage of intermediate buffering is it reduces the overall buffer space required [Ref. 21], and helps to alleviate packet bunching and hence congestion [Ref. 22].

The final two approaches do not depend on buffering at intermediate nodes to help reduce jitter. As a result, the receiver needs to provide all of the necessary buffering. This is only now becoming realistic due to the falling costs of memory [Ref. 23]. The advantages of this method are that the routers do not need to be bothered with additional buffering. This will facilitate processing and thus keep processing time at the router minimal. On the otherhand congestion dictates the need to implement pre-emptive buffering to prevent loss of real-time traffic packets.

A. Stop and Go Queueing

Golestani proposes Stop and Go queuing [Ref. 22, 24, 25, 26]. This method entails the use of a time based framing strategy. The source declares the maximum bit rate r_k over a period T . A channel will not be established if the sum of all r_k

over any link exceeds the capacity of that link. The period, T , is also involved in the framing strategy. As the packets are transmitted from the source they are placed in a time frame of size T . At the next node, none of the packets in this time frame are eligible for transmission until the frame is received in its entirety. Once the frame is received, all of the packets must be transmitted in the next frame of size T that is leaving that node on the desired link. A disadvantage of this technique is that there is a requirement of initial synchronization between adjacent nodes. This is needed so that a frame transmitted is identical to the frame received at the next node. Also the clocks at each node need to create each T exactly the same size, otherwise it is possible for packets starting out in the same frame to get split up between frames.

B. Tenet Group

The Tenet group at UC Berkeley has put forth a series of proposals [Ref. 27, 28, 21]. The latest is called Rate-Controlled Static-Priority Queueing. This technique uses an admission control and a two part server. The server consists of a rate controller and a scheduler. The rate controller can be set up to control rate-jitter or delay jitter. The rate-jitter controller ensures the traffic pattern for each source at any node complies with the traffic characterization submitted during the admission control. The delay-jitter controller fully reconstructs the traffic pattern as sourced at each router. Regardless of the specific type, the rate controller assigns an eligibility time to each packet. The packet is not sent to the scheduler until after its eligibility time. The Scheduler consists of several priority queues and a non real-time packet queue. All of the queues are FCFS. Each queue level corresponds to a different delay bound, the highest priority having the smallest delay bound. The server then sends the first packet from the highest non-empty priority queue.

C. Predictive Service

Predictive Service [Ref. 8, 29] utilizes a playback point. In order to calculate the playback point requires knowledge of the bound on delay and an estimate of the percentage of packets missing this bound. Those applications that require very low loss rates will use the network computed maximum delay bound. Other services that prefer to see less delay at the expense of higher losses will set the playback point based on the delay actually experienced on the network plus some additional time to allow for expected jitter.

For traffic with rigid requirements in delay and reliability a scheme employing Weighted Fair Queuing is proposed. This method is based on the maximum bounded delay as a result of a flow receiving the same clock rate at every switch. This will be the network's calculated delay. Modeled on a leaky bucket scheme the bound on the delay is b/r where b is the bucket depth for the flow and r is the rate at which the bucket fills up and also is the minimum bandwidth required. This method has the disadvantage of tying the bandwidth requirement to the delay.

For traffic that can tolerate some loss due to routing at the benefit of less cost and less delay, a mechanism called Predictive Service is presented. A key part to this service is a queuing discipline called FIFO+. FIFO+ attempts to make resource sharing fair over multiple hops and not just for a single hop. For this the packets are placed in the transmission queue according to the time they should have received service. This is determined by the average delays according to the packet classification. If all packets in a given class receive the average service then the queue will appear to simply be FIFO.

D. RSVP

RSVP (ReSerVation Protocol) [Ref. 6] is a protocol layer that utilizes the underlying routing protocol. It is only concerned with making bandwidth reservations and is not intimately involved in making guarantees. It would be used by a layer that determines the necessary reservation to meet the quality of service requirements. It utilizes receiver initiated reservations so as to allow for heterogenous receiver types. Also it implements the use of filters. A filter is set to allow only a receiver specified subset of senders to utilize the reserved resources. This permits the receiver to keep the reserved resources but provides the ability to change the source(s) received over those resources.

RSVP only attempts to make reservation along a pre-existing path. If the resources are not available along the path chosen by the routing protocol, then the receiver will have to settle for best effort. There is no attempt made to have the routing protocol find a path along which sufficient reservations can be made.

VIII. Concluding Remarks

This thesis addressed the problem of constructing multicast trees with guaranteed QoS that utilize the network resources efficiently. It identified design goals for constructing such trees and presented an integrated approach to achieve an efficient combination of CSTs and SSTs based on the *a priori* information about the participants. The specific contributions made are as follows:

- A systematic approach, based on the critical participants, is described to select a combination of CSTs and SSTs for an interaction.
- A scalable center location mechanism and protocol, that selects a center transparently in a distributed fashion and balances the network resource consumption, is described.
- It is shown, by simulation modeling, that CSTs based on center location as above, provide almost the same delay as SSTs.
- It is shown, by simulation modeling, that CSTs are efficient when the number of concurrent senders is small as compared to the number of participants.
- It is also shown that CSTs with centers located as above compare very well with Steiner trees.

A. Suggestions for Future Research

The approach proposed in this thesis is superior in many respects to the PIM approach currently being considered by the Internet Engineering Task Force for standardization [Ref. 13]. However, to reach the same level of maturity and to add insight

to tree construction mechanisms required for wide-area multicasting with minimum QoS in the presence of a large number of interactions, the following issues need to be addressed.

- A center selection mechanism that permits asymmetric link costs needs to be developed. This extension will make this approach the most general of all the approaches proposed so far in the literature.
- Selecting a hierarchy of distribution centers will make this approach scalable to very large, widely distributed interactions with a large number of senders. In the near future, distributed interactive simulation is expected to require such a capability. The concepts of distribution centers proposed here and the rendezvous points proposed in PIM (which can be looked upon as reception centers) could be combined in the most general case.
- One question that needs to be answered by additional simulations is: how effective is the proposed center location protocol in balancing the network load in the presence of multiple interactions?
- In the proposed approach, no provision is made for reconfiguring the distribution centers. It is assumed that the number of unanticipated senders of an interaction does not change so excessively during the life of an interaction that the distribution centers selected by the SR are no longer lead to efficient use of network resources. This is a difficult question to answer since entirely new CSTs will have to be created in such a case. The answer probably lies in selecting new centers periodically.
- Detailed specifications of the registration protocol, the tree construction protocol, and the reservation protocol need to be developed. In particular, the

attributes registered and their use to determine the CSPs of an interaction need to be categorically identified.

- It is expected that the quality of the CSTs resulting from the proposed mechanism be relatively insensitive to the initial pairing. This needs to be verified with additional simulations.

APPENDIX A

PROGRAM CODE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               THESISGRAPH.M                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The graph generator portion of this program is based on RG2 as          %
% described by W. M. Waxman in his article "Routing of Multipoint         %
% Connections" in IEEE Journal on Selected Areas in Communications,      %
% December 1988.                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                           November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% xcor and ycor are dimensions for locating the nodes                    %
% n = number of nodes to be represented                                  %
% alpha and beta are parameters of edge descriptions                     %
% l (el) is the max distance between any pair of nodes.                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global sp d n hops

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The following command starts the script file that asks the user for    %
% the parameters alpha, beta, n, and l and determines the values of      %
% xcor and ycor.                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

get_vals

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The next command is a script file that determines the existence       %
% of edges between any two nodes and the cost to use that edge. It      %
% generates the matrix "d[n,n]" that is a table that contains the cost  %
% between any two nodes.                                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

edges

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The next command is a script file that plots the graph. The values %
% required for the plot are generated in edges.m. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

plt_grph

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The next command is a script file that asks for the number of %
% critical participants and randomly locates them among the nodes. The %
% node address of each cp is kept in the vector csp[cp]. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

get_cps

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The following command initialize 4 matrices. sp is the distance to %
% the column node on the shortest path tree sourced at the row node, %
% stein is the distance to the column node with the row node being the %
% source of the steiner tree. hops and hopstein are the intermediate %
% nodes between the source and destination node for sp and stein %
% respectively. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
hops = zeros(n*n , n-2);
sp = -ones(n, n);
hopstein = zeros(cp^2 , cp-2);
stein = -ones(cp, cp);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The next command is a script file that finds the core location which %
% is stored in the variable "core" as the address of the node selected. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

findcore

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Next, the adjacency matrix where, the only edges are the branches of %
% the tree, is made. This enables to determine the distance between %
% CP's on the tree. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

if sp(core,1) == -1
    [sp(core, :), hops((core-1)*n+1:core*n,:)] = shrtpath(core, d);
end
d_cor = tree(core, csp, hops((core-1)*n + csp, :), d);

[sparse, sparshop, root] = sparsest(sp, hops, csp);
d_sparse = tree(root, csp, sparshop(csp, :), d);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The next command is a script file that calculates the average path %
% length for each type of tree and the cost for each type of tree %
% depending on the number of simultaneously transmitting senders. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

gen_data

out_data

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               GET_VALS.M                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is a script file that obtains the parameters required for      %
% thesigraph.m                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                         November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% get alpha %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

while 1
    fprintf('\nalpha is a relative value (0,1]')
    fprintf('\nwhere a small alpha will increase')
    fprintf('\nthe number of short edges relative')
    fprintf('\nto the number of long edges.\n')
    alpha = input('Enter the value for alpha: ');
    if alpha > 0
        if alpha <= 1
            break
        end
    end
    fprintf('\n\nalpha needs to be > 0 and <= 1.\n')
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% get beta %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

while 1
    fprintf('\n\nbeta is a relative value (0,1]')
    fprintf('\nthat is proportional to the number')
    fprintf('\nof edges.\n')
    beta = input('Enter the value for beta: ');
    if beta > 0
        if beta <= 1
            break
        end
    end
    fprintf('\n\nbeta needs to be > 0 and <= 1.\n')
end

```

```

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% get n and determine row and col dimensions %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

while 1
    n = input('Enter the number of nodes: ');
    if n > 0
        if n == round(n)
            break
        end
    end
    fprintf('\n\nThe number of nodes must be a positive integer.\n')
end
xcor = ceil(4*sqrt(n/3));
ycor = ceil(3*sqrt(n/3));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% get maximum distance l (el) %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

while 1
    l = input('Enter the maximum cost between nodes: ');
    if l > 0
        if l == round(l)
            break
        end
    end
    fprintf('\n\nThe max cost needs to be entered as a positive integer.\n')
end

```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               EDGES.M                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This script file is part of thesgraph.m                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% It determines the edges and their cost between nodes and generates %
% information required for the plot of the graph.                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                         November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% determine cost between pairs                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

d=zeros(n,n);

for i=1:n-1
    for j=i+1:n
        d(i,j) = ceil(l*rand);
    end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Prepare for values needed to make a plot.                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The node sequence locates the nodes on a rectangular grid.         %
% xcor and ycor are generated and n is obtained in the script file    %
% get_vals.m                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

countedg=0;
edgex=[];
edgex=[];
cost=[];
countn=n;

node=zeros(n,2);
coordlef = xcor*ycor;
for j=1:xcor

```

```

    for k=1:ycor
        if rand <= countn / coordlef
            node(countn,:)= [j,k];
            countn=countn-1;

        end
        coordlef=coordlef-1;
        if n==0
            break,break
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% determine existence of edges, note a value of Inf means no edge %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:n-1
    for j=i+1:n
        if rand > beta*exp(-d(i,j)/(1*alpha))
            d(i,j)=Inf;
        else
            countedg=countedg+1;
            cost=[cost; d(i,j)];
            y = node(j,2) - node(i,2);
            x = node(j,1) - node(i,1);
            edgex = [edgex; node(i,1), ...
                    node(i,1) + x/2 + (abs(y)-1)/(xcor-1), node(j,1)];
            edgey=[edgey; node(i,2), ...
                    node(i,2) + y/2 + (abs(x)-1)/(ycor-1), node(j,2)];
        end
        d(j,i) = d(i,j);
    end
end

end
avg_deg = (sum(sum(finite(d)))-n)/(n-1);
fprintf('\nThe average node degree is %f\n',avg_deg)

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               PLT_GRP.H.M                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This script file is part of thesisgraph.m.                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% edgex, edgey, cost, countedg, and node generated in the script file    %
% edges.m.                                                                %
% alpha, beta, n, l are obtained in the script file get_vals.m.         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code prints the mapping of node connections and asks if it is    %
% desired to have the nodes and edges labeled. The default is n (no).    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                           November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

tog=input('Do you desire the nodes labeled and edge costs printed? [y,n]:','s');

```

```

hold off

```

```

clg

```

```

hold on

```

```

for i=1:n

```

```

    plot(node(i,1),node(i,2),'wo')

```

```

    if tog == 'y'

```

```

        text(node(i,1),node(i,2),[' ',setstr(i+64)])

```

```

    end

```

```

end

```

```

for i=1:countedg

```

```

    plot(edgex(i,:),edgey(i,:),'w-')

```

```

    if tog == 'y'

```

```

        text(edgex(i,2),edgey(i,2),int2str(cost(i)))

```

```

    end

```

```

end

```

```

title(['alpha = ', num2str(alpha), '    beta = ', num2str(beta), ...

```

```

        '    max cost = ', int2str(l), '    nodes = ', int2str(n)])

```

```

ylabel(['avg node deg = ',num2str(avg_deg)])

```

```

axis off

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               GET_CPS.M                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This script file is part of thesisgraph.m                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This script file asks for the desired number of Critical Participants %
% and then randomly locates them among the nodes                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                         November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

while 1
    cp = input('Enter the number of Critical Participants: ');
    if cp > 0
        if cp <= n
            if cp == round(cp)
                break
            end
        end
    end
    fprintf('\n\nThe number of critical participants')
    fprintf('\nmust be a positive integer <= the number')
    fprintf('\nof nodes which is %g.\n\n',n)
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% locate the CP's randomly %
% The vector csp[cp] keeps the node address of each CP %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

cpcount = cp;
nodelef = n;
csp = zeros(1,cp);

for i = 1:n
    if rand <= cpcount/nodelef
        csp(cpcount) = i;
        cpcount = cpcount - 1;
    end
    nodelef = nodelef - 1;
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                FINDCORE.M                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is a script file for thesisgraph.m                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This script file will locate the core. It uses the Critical Set of    %
% Participants and finds the optimal core placement. It first pairs the %
% CP's by putting the most distant CP's together. Each pair finds the %
% intermediate node closest to the midpoint between them. This node is %
% then paired with the midpoint of another pair. This process is        %
% continued until a single node is selected.                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                         November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 1:cp
    a = csp(i);
    [sp(a,:), hops((a-1)*n+1 : a*n, :)] = shrtpath(a, d);
    if i == 1
        if sum(sum(isinf(sp(a,csp)))) > 0
            error('The CPs are not fully connected, try again.')
        else
            fprintf('\nIts good!!\n')
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DETERMINE THE PAIRS %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dcp = sp(csp,csp);
cptally = csp;
pairs = [];
cpcount = cp;

while cpcount > 2
    [m, ind1] = max(dcp);
    [m, ind2] = max(m);
    pairs = [pairs, cptally(ind1(ind2)), cptally(ind2)];
    cptally([ind1(ind2) ind2]) = [];
    dcp([ind1(ind2) ind2],:) = [];
    dcp(:, [ind1(ind2) ind2]) = [];
end

```

```

        cpcount = cpcount - 2;
end
pairs = [pairs, cptally];

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This portion of code will properly locate the bytes %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% tourn will keep the addresses of the nodes involved %
% and a value of -1 is the location of a byte          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

k = ceil(log2(cp));
bye = 2^k - cp;
countb = 0;
tourn = zeros(k+1,2^k);

```

```

while bye > 0
    m = 2^floor(log2(bye));
    for i = 1:m
        tourn(1,2^(k-countb)-m+i) = -1;
    end
    bye = bye - m;
    countb = countb + 1;
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This portion of code loads the pairs into the matrix %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% the vector "pairs" lists the addresses of the cp's as %
% they are paired, the odd one out will be at the end %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

countp = 0;

```

```

for i=1:2:2^k-1
    if tourn(1,i) ~= -1                % check if paired bye
        if tourn(1,i+1) ~= -1        % check if single bye
            for j=1:2
                tourn(1,i+j-1) = pairs(countp+j);
            end
            countp = countp + 2;
        else

```

```

                                tourn(1,i) = pairs(cp);
                                end
                                end
                                end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This portion of code will complete the selection %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for phase = 1:k
    for i = 1:2^(k - phase + 1) - 1
        if tourn(phase,i+1) == -1 | tourn(phase,i+1) == tourn(phase,i)
            tourn(phase+1,(i+1)/2) = tourn(phase,i);
        else
            tourn(phase+1,(i+1)/2) = findcent(tourn(phase,i), ...
                                                tourn(phase,i+1));
        end
    end
end

core = tourn(k+1,1);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                GEN_DATA.M                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This script file is for use in thesisgraph.m                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This script file will calculate the average path length for the core %
% based tree, the shortest path (source based tree) and the steiner    %
% tree. It will also calculate tree costs.                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                         November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

cor_pth_len = [];
sp_len = [];
sparse_len = [];
cor_hops = zeros(cp*n, n-2);
spars_hops = zeros(cp*n, n-2);

for i = 1:cp
    [coredist, cor_hops((i - 1)*n + 1 : i*n, :)] = ...
        shrtpath(csp(i), d_cor);
    cor_pth_len = [cor_pth_len; coredist(csp)];
    sp_len = [sp_len; sp(csp(i),csp)];
    [spardist, spars_hops((i - 1)*n + 1 : i*n, :)] = ...
        shrtpath(csp(i), d_sparse);
    sparse_len = [sparse_len; spardist(csp)];
end
avg_cor_pl = sum(cor_pth_len)/(cp-1);
avg_sh_pl = sum(sp_len)/(cp-1);
avg_sprs_pl = sum(sparse_len)/(cp-1);

cor_avg = sum(avg_cor_pl)/cp;
sh_avg = sum(avg_sh_pl)/cp;
sprs_avg = sum(avg_sprs_pl)/cp;

hops_corin = [];
hops_shin = [];
hops_sprsin = [];

for i = 1:cp
    hops_corin = [hops_corin; cor_hops((i - 1)*n + csp, :)];
    hops_shin = [hops_shin; hops((csp(i) - 1)*n + csp, :)];

```



```

        hops_sprsin = [hops_sprsin; spars_hops((i-1)*n + csp, :)];
end

cor_max = max_use(csp, hops_corin);
sh_max = max_use(csp, hops_shin);
sprs_max = max_use(csp, hops_sprsin);

crcst = [];
shcst = [];
sprscst = [];

for i = 1:cp - 1
    crcst = [crcst, tre_cost(cor_max, i, Inf2zero(d))];
    shcst = [shcst, tre_cost(sh_max, i, Inf2zero(d))];
    sprscst = [sprscst, tre_cost(sprs_max, i, Inf2zero(d))];
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                OUT_DATA.M                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This script file is for use in thesisgraph.m                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This file will output the data generated to graphs for analysis.      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                           November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

input('Which server printer do you want? [2,3,4,5] : ');
printer = ['print1',int2str(r)];
xlabel(['avg path len: CBT = ',num2str(cor_avg),' SBT = ', ...
        num2str(sh_avg),' MST = ',num2str(spars_avg)])
eval(printer)
hold off
clg
avg_pl = [sh_avg, cor_avg, spars_avg];
plot(1:3,avg_pl,'*')
title('Average Path Length')
grid
eval(printer)
clg
hold on
for i = 1:cp - 1
    plot(i, crcst(i),'x')
    plot(i, shcst(i),'o')
    plot(i, sprscst(i),'+')
end
grid
title('TREE COST vs # SIMULTANEOUS SENDERS')
xlabel('x = CBT, o = SBT, + = sparse')
eval(printer)

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                FINDCENT.M                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function is required by thesisgraph.m                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      y = findcent(a,b)      a and b are the nodes between which the %
% intermediate node closest to the center is desired. The GLOBAL      %
% variables are the matrix of sp[n,n] which gives the shortest path   %
% length between any two nodes; the matrix hops gives the sequence of %
% nodes between those two nodes; n is the total number of nodes; and d %
% is the adjacency matrix for all nodes.                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                         November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function y = findcent(a,b)

global sp d hops n

if sp(a,b) == -1
    [sp(a,:), hops((a-1)*n+1 : a*n, :)] = shrtpath(a, d);
end
midcost = sp(a,b)/2;
rowab = (a-1)*n+b;
count = 1;
sumcost = 0;

while sumcost < midcost
    if hops(rowab,count) == 0
        lhop = b;
    else
        lhop = hops(rowab,count);
    end
    sumcost = sp(a, lhop);
    count = count + 1;
end
count = count - 2;
if count == 0
    fhop = a;
else
    fhop = hops(rowab,count);
end

if sumcost - midcost == d(fhop,lhop)/2

```

```
        y = min(fhop,lhop);  
elseif sumcost - midcost > d(fhop,lhop)/2  
    y = fhop;  
else  
    y = lhop;  
end
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Inf2zero                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function is used in theisgraph.m. Its primary use is to change %
% the Inf in a adjacency matrix modified by tree.m to zeros so that the %
% cost of the tree can be determined.                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       out = Inf2zero(in)       where in is a matrix of any size and %
% out is the same input matrix but with any Inf's changed to 0.      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                     November 1993    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function out = Inf2zero(in)

[row, col] = size(in);

for i = 1:row
    for j = 1:col
        if in(i,j) == Inf
            out(i,j) = 0;
        else
            out(i,j) = in(i,j);
        end
    end
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                LAST_HOP.M                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function is for use in thesisgraph.m                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function will find the last column of a vector that does not      %
% contain a padded zero.                                                  %
%       out = last_hop(in_hops)      where in_hops is the input          %
% vector and out will be the column number.                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                           November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function out = last_hop(in_hops);

```

```

    out = length(in_hops);
    while in_hops(out) == 0
        out = out - 1;
        if out == 0
            break
        end
    end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                     max_use.m                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function is for use in thesisgraph.m                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function will determine the number of sources that use each hop. %
%       cnt_out = max_use(concern, branch)       where concern is a %
% vector listing the nodes between which the count is desired. It is %
% assumed that the nodes in this vector are the sources and destina- %
% tions. branch[length(concern)^2,n-2] is a matrix that lists the %
% intermediate hops between a node and its destinations. Note that n is %
% the number of nodes in the original graph. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                     November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function cnt_out = max_use(concern, branch)

```

```

n = length(branch(1,:)) + 2;
m = length(concern);
cnt_out = zeros(n,n);

```

```

for i = 1:m
    sub_cnt = zeros(n,n);
    for j = 1:m
        if i ~= j
            col = last_hop(branch((i-1)*m + j, :));
            cnt_to = concern(j);
            if col == 0
                cnt_fm = concern(i);
            else
                cnt_fm = branch((i-1)*m + j,col);
            end
            while sub_cnt(cnt_fm,cnt_to) == 0
                sub_cnt(cnt_fm,cnt_to) = 1;
                col = col - 1;
                if col == -1
                    break
                end
                cnt_to = cnt_fm;
                if col == 0
                    cnt_fm = concern(i);
                else
                    cnt_fm = branch((i-1)*m + j,col);
                end
            end
        end
    end
end

```

```
                                end
                        end
                end
        end
        cnt_out = cnt_out + sub_cnt;
end
```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                SHRTPATH.M                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function is for use in thesisgraph.m.                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      [sp, hops] = shrtpath(a, d)                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% shrtpath input will be source node address (a) and the adjacency      %
% matrix d[n,n]. Output will be sp[n] which is a vector of distances    %
% from the source to all other nodes and hops[n,n-2] which is the      %
% intermediate nodes between the source and the destination. Note that %
% hops is padded with zeros to make all row vectors have a length      %
% of n-2.                                                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                           November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function [sp, hops] = shrtpath(a, d)

n = size(d,1);

sp=Inf*ones(1,n);
parent = a*ones(1,n);
hops = [];

sp(a) = 0;
ntally = 1:n;

while length(ntally) > 1
    [m,i] = min(sp(ntally));
    checkwho = ntally(i);
    ntally(i) = [];
    dsp = d(checkwho,ntally);
    for i = 1:length(ntally)
        if m + dsp(i) < sp(ntally(i))
            sp(ntally(i)) = m + dsp(i);
            parent(ntally(i)) = checkwho;
        end
    end
end

for j = 1:n
    hopping = parent(j);
    while hopping(1) ~= a
        hopping = [parent(hopping(1)), hopping];
    end
end

```

```
end
hopping(1) = [];
hopping = [hopping, zeros(1, n-2 - length(hopping))];
hops = [hops; hopping];
end
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               SPARSEST.M                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function is for use in thesisgraph.m                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       [sparse, sparshop, root] = sparsest(sp, hops, csp)              %
% where sp is the shortest distance between all nodes of graph, hops is %
% the intermediate hops between any two nodes of the graph, and csp is  %
% a vector that contains the nodes that need to be connected by the     %
% sparse steiner tree. The output sparse, is the shortest distance from  %
% the node in csp selected as root and all other nodes. If the         %
% destination node is not on the sparse tree, then that value will be   %
% Inf; root is the node in csp selected as the root of the tree.       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                         November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function [sparse, sparshop, root] = sparsest(sp, hops, csp)

n = size(sp,1);
sparse = Inf*ones(1,n);
sparshop = zeros(n, n-2);
dstein = sp(csp,csp);
cp = length(csp);

for i = 1:cp
    dstein(i,i) = Inf;
end

[m,i] = min(dstein);
[m,j] = min(m);
root = csp(i(j));
col = last_hop(hops((root-1)*n+csp(j),:));
ntally = csp;
ntally([i(j) j]) = [];
mtally = [root, hops((root-1)*n+csp(j),1:col), csp(j)];
sparshop(csp(j),:) = hops((root-1)*n+csp(j),:);
sparse(csp(j)) = sp(root,csp(j));
for k = 1:col
    if k ~= 1
        sparshop(sparshop(csp(j), k),1:k-1) = sparshop(csp(j), 1:k-1);
    end
    sparse(sparshop(csp(j), k)) = sp(root, sparshop(csp(j), k));
end

```

```

end
while length(ntally) > 1
    dstein = sp(ntally, mtally);
    [m,i] = min(dstein);
    [m,j] = min(m);
    if mtally(j) ~= root
        col1 = last_hop(sparshop(mtally(j), :));
        part1 = [sparshop(mtally(j), 1:col1), mtally(j)];
    else
        part1 = [];
    end
    col1 = length(part1);
    col2 = last_hop(hops((ntally(i(j)) - 1)*n + mtally(j), :));
    part2 = fliplr(hops((ntally(i(j)) - 1)*n + mtally(j), 1:col2));
    if part1 ~= [] | part2 ~= []
        sparshop(ntally(i(j)), 1:(col1 + col2)) = ...
            [part1, part2];
    end
    sparse(ntally(i(j))) = sparse(mtally(j)) + ...
        sp(ntally(i(j)), mtally(j));
    for x = 1:col2
        if col1 ~= 0 | x > 2
            sparshop( part2(x) , 1:col1+x-1 ) = ...
                sparshop(ntally(i(j)), 1:col1+x-1);
        end
        sparse(part2(x)) = sparse(ntally(i(j))) - ...
            sp(ntally(i(j)), part2(x));
    end
    ntally = [mtally, part2, ntally(i(j))];
    ntally(i(j)) = [];
end
dstein = sp(ntally,mtally);
[m,j] = min(dstein);
if mtally(j) ~= root
    col1 = last_hop(sparshop(mtally(j),:));
    part1 = [sparshop(mtally(j),1:col1), mtally(j)];
else
    part1 = [];
end
col1 = length(part1);
col2 = last_hop(hops((ntally-1)*n+mtally(j),:));
part2 = fliplr(hops((ntally-1)*n+mtally(j),1:col2));
if part1 ~= [] | part2 ~= []
    sparshop(ntally,1:(col1 + col2)) = [part1, part2];
end

```

```

end
sparse(ntally) = sparse(mtally(j)) + sp(ntally,mtally(j));
for x = 1:col2
    if col1 ~= 0 | x > 2
        sparshop(part2(x), 1:(col1+x-1)) = sparshop(ntally, ...
            1:(col1+x-1));
    end
    sparse(part2(x)) = sparse(ntally) - sp(ntally, part2(x));
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                TRE_COST.M                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function is for use with theisgraph.m                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function will determine the cost of a tree given a matrix of the %
% number of times a tree uses a hop between two nodes. This matrix can %
% be generated by the function max_use. The function also requires the %
% number of simultaneously transmitting sources, and finally an %
% adjacency matrix that indicates the cost between nodes. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                out = tree_cost(maxmat, val, d)           %
%                                where maxmat is %                         %
% the matrix indicating the number of times a hop is used; val is the %
% number of simultaneously transmitting senders; and d is the adjacency %
% matrix with the cost of adjacent hops. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                           November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function out = tre_cost(maxmat, val, d)

```

```

most = val*ones(size(maxmat));
least = min(most, maxmat);
out = sum(sum(least .* d));

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                TREE.M                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function is for use in thesisgraph.m                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function will determine the adjacency matrix for a given tree. %
%      d_tree = tree(src, concern, branch[(length(concern),n-2], d) %
% where src = node id of the root, concern = vector of node ids to   %
% which the tree is to be made, branch is the intermediate nodes     %
% between the root and the nodes listed in concern. d is the adjacency %
% matrix of the original graph.                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eric B. Boyer                                                    November 1993 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function d_tree = tree(src, concern, branch, d)

n = length(branch(1,:)) + 2;
d_tree = Inf*ones(n,n);
for i = 1:n
    d_tree(i,i) = 0;
end

for i = 1:length(concern)
    if concern(i) ~= src
        col = last_hop(branch(i, :));
        d_to = concern(i);
        if col == 0;
            d_fm = src;
        else
            d_fm = branch(i,col);
        end
        while d_tree(d_fm,d_to) == Inf
            d_tree(d_fm,d_to) = d(d_fm,d_to);
            d_tree(d_to,d_fm) = d(d_to,d_fm);
            col = col - 1;
            if col == -1
                break
            end
            d_to = d_fm;
            if col == 0;
                d_fm = src;
            else
                d_fm = branch(i,col);
            end
        end
    end
end

```

end
end
end
end

LIST OF REFERENCES

- [1] Stephen E. Deering and David R. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Transactions on Computer Systems*, May 1990.
- [2] Tony Ballardie, Paul Francis, and Jon Crowcroft, "Core based trees (CBT) an architecture for scalable inter-domain multicast routing," in *ACM SIGCOMM*, September 1993.
- [3] J. Moy, *Multicast extensions to OSPF*, Internet Draft of the Network Working Group, July 1993.
- [4] D. Waitzman, C. Partridge, and S. Deering, *Distance vector multicast routing protocol*, Technical Report RFC 1075, Internet, Network Working Group, November 1988.
- [5] Liming Wei and Deborah Estrin, *A comparison of multicast trees and algorithms*, Draft submitted to INFOCOM 1994, 1993.
- [6] Lixia Zhang, Stephen E. Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala, "RSVP: a new resource ReSerVation Protocol," *IEEE Networks Magazine*, September 1993.
- [7] Domenico Ferrari, "Client requirements for real-time communication services," *IEEE Communications Magazine*, November 1990.
- [8] David D. Clark, Scott Shenker, and Lixia Zhang, "Supporting real-time applications in an integrated services packet network: Architecture and mechanism," in *ACM SIGCOMM*, September 1992.
- [9] Mark Moran and Bernd Wolfinger, *Design of a continuous media data transport service and protocol*, Technical Report TR-92-019, Tenet Group, International Computer Science Institute, 1992.
- [10] H. Schulzrinne, *Issues in designing a transport protocol for audio and video conferences and other multiparticipant real-time applications*, Internet Draft of the Audio-Video Transport Working Group, October 1993.
- [11] Steven Randall Zeswitz, *NPSNET: integration of distributed interactive simulation (DIS) protocol for communication architecture and information interchange*, Master's thesis, Naval Postgraduate School, September 1993.

- [12] Naval Ocean Systems Center, *Navy UHF Satellite Communication System Description*, 1991.
- [13] S. Deering, D. Estrin, D. Farinacci, and V. Jacobson, *IGMP router extensions for routing to sparse multicast-groups*, September 1993.
- [14] C. Hedrick, *Routing Information Protocol*, Technical Report RFC 1058, Internet, Network Working Group, June 1988.
- [15] Stephen E. Deering, *Host extensions for IP multicasting*, Technical Report RFC 1112, Internet, Network Working Group, August 1989.
- [16] J. Moy, *OSPF version 2*, Technical Report RFC 1247, Internet, Network Working Group, July 1991.
- [17] Van Jacobson, *The Session Directory Tool*, Lawrence Berkeley Laboratory, 1992.
- [18] Matthew Doar and Ian Leslie, "How bad is naive multicast routing?" in *Proceedings of INFOCOM*, March 1993.
- [19] S. Zabele and R. Braudes, *Requirements for multicast protocols*, Technical Report RFC 1458, Internet, Network Working Group, May 1993.
- [20] Bernard M. Waxman, "Routing of multipoint connections," *IEEE Selected Areas in Communications*, December 1988.
- [21] Dinesh C. Verma, Hui Zhang, and Domenico Ferrari, "Delay jitter control for real-time communication in a packet switching network," in *Proc. of TriComm 1991*, April 1991.
- [22] S. Jamaloddin Golestani, "Congestion-free transmission of real-time traffic in packet networks," in *Proceedings of INFOCOM*, June 1990.
- [23] Garret A. Wollman, notes from Van Jacobson talk of 24 June 1993, presented over MBONE.
- [24] S. Jamaloddin Golestani, "Duration-limited statistical multiplexing of delay-sensitive traffic in packet networks," in *Proceedings of INFOCOM*, April 1991.
- [25] S. Jamaloddin Golestani, "A framing strategy for congestion management," *IEEE Selected Areas in Communications*, September 1991.
- [26] S. Jamaloddin Golestani, "Congestion-free communication in high-speed packet networks," *IEEE Transactions on Communications*, December 1991.

- [27] Hui Zhang and Domenico Ferrari, *Rate-controlled static-priority queueing*, Technical Report TR-92-003, Computer Science Division, University of California at Berkeley, February 1992.
- [28] Domenico Ferrari, *Real-time communication in an internetwork*, Technical Report TR-91-001, Tenet Group, International Computer Science Institute, 1991.
- [29] Sugih Jamin, Scott Shenker, Lixia Zhang, and David D. Clark, *An admission control algorithm for predictive real-time service (extended abstract)*, Preliminary Draft, Xerox Palo Alto Research Center.

BIBLIOGRAPHY

Thomas E. Anderson, Susan S. Owicki, James B. Saxe, and Charles P. Thacker, "High speed switch scheduling for local area networks," *ACM SIG PLAN Notices*, September 1992.

K. Bala, I. Cidon, and K. Sohraby, "Congestion control for high speed packet switched networks," in *Proceedings of INFOCOM*, June 1990.

Tony Ballardie, Paul Francis, and Jon Crowcroft, "Core based trees (CBT) an architecture for scalable inter-domain multicast routing," in *ACM SIGCOMM*, September 1993.

Anindo Banerjea and Srinivasan Keshav, *Queueing delays in rate controlled networks*, Technical Report TR-92-015, Tenet Group, International Computer Science Institute, 1992.

Nasr E. Belkeir and Mustaque Ahamad, "Low cost algorithms for message delivery in dynamic multicast groups," in *International Conference on Distributed Computing*, pages 73-80, June 1989.

C. Topolcic, editor, *Experimental internet stream protocol, version 2 (ST-II)*, Technical Report RFC 1190, Internet, Network Working Group, October 1990.

David D. Clark, Scott Shenker, and Lixia Zhang, "Supporting real-time applications in an integrated services packet network: Architecture and mechanism," in *ACM SIGCOMM*, September 1992.

S. Deering, D. Estrin, D. Farinacci, and V. Jacobson, *IGMP router extensions for routing to sparse multicast-groups*, September 1993.

Stephen E. Deering, *Host extensions for IP multicasting*, Technical Report RFC 1112, Internet, Network Working Group, August 1989.

Stephen E. Deering, "MBONE - the multicast backbone (videoconferencing over the internet)," CERFnet Seminar, March 1993.

Stephen E. Deering and David R. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Transactions on Computer Systems*, May 1990.

Matthew Doar and Ian Leslie, "How bad is naive multicast routing?" in *Proceedings of INFOCOM*, March 1993.

Domenico Ferrari, "Client requirements for real-time communication services," *IEEE Communications Magazine*, November 1990.

Domenico Ferrari, *Real-time communication in an internetwork*, Technical Report TR-91-001, Tenet Group, International Computer Science Institute, 1991.

Ariel J. Frank, Larry D. Wittie, and Arthur J. Bernstein, "Multicast communications on network computers," *IEEE Software*, 1985.

S. Jamaloddin Golestani, "Congestion-free transmission of real-time traffic in packet networks," in *Proceedings of INFOCOM*, June 1990.

S. Jamaloddin Golestani, "Congestion-free communication in high-speed packet networks," *IEEE Transactions on Communications*, December 1991.

S. Jamaloddin Golestani, "Duration-limited statistical multiplexing of delay-sensitive traffic in packet networks," in *Proceedings of INFOCOM*, April 1991.

S. Jamaloddin Golestani, "A framing strategy for congestion management," *IEEE Selected Areas in Communications*, September 1991.

Amit Gupta and Mark Moran, *Channel groups, a unifying abstraction for specifying inter-stream relationships*, Technical Report TR-93-015, Computer Science Division, University of California at Berkeley, March 1993.

C. Hedrick, *Routing Information Protocol*, Technical Report RFC 1058, Internet, Network Working Group, June 1988.

Van Jacobson, *The Session Directory Tool*, Lawrence Berkeley Laboratory, 1992.

Sugih Jamin, Scott Shenker, Lixia Zhang, and David D. Clark, *An admission control algorithm for predictive real-time service (extended abstract)*, Preliminary Draft, Xerox Palo Alto Research Center.

Vachaspathi P. Kompella, Joseph C. Pasquale, and George C. Polyzos, "Multicasting for multimedia applications," in *Proceedings of INFOCOM*, May 1992.

Mark Moran and Bernd Wolfinger, *Design of a continuous media data transport service and protocol*, Technical Report TR-92-019, Tenet Group, International Computer Science Institute, 1992.

J. Moy, *OSPF version 2*, Technical Report RFC 1247, Internet, Network Working Group, July 1991.

J. Moy, *Multicast extensions to OSPF*, Internet Draft of the Network Working Group, July 1993.

Naval Ocean Systems Center, *Navy UHF Satellite Communication System Description*, 1991.

C. Partridge, *A proposed flow specification*, Technical Report RFC 1363, Internet, Network Working Group, September 1992.

J. Pasquale, G. Polyzos, E. Anderson, and V. Kompella, "The multimedia multicast channel," in *Proceedings of the 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, November 1992.

Joseph C. Pasquale, George C. Polyzos, and Vachaspathi P. Kompella, *The multimedia multicasting problem*, Technical report, Department of Computer Science and Engineering, University of California, San Diego, October 1992.

Srinivas Ramanathan, P. Venkat Rangan, Harrick M. Vin, and Thomas Kaeppner, "Optimal communication architectures for multimedia conferencing in distributed systems," in *International Conference on Distributed Computing Systems*, June 1992.

P. Venkat Rangan, Srinivas Ramanathan, Harrick M. Vin, and Thomas Kaeppner, "Techniques for multimedia synchronization in network file systems," *Computer Communications Journal*, March 1993.

P. Venkat Rangan and Harrick M. Vin, *Multimedia Conferencing as a Universal Paradigm for Collaboration*, chapter 14, Springer-Verlag, 1991, Edited by Lars Kjeldahl.

P. Venkat Rangan, Harrick M. Vin, and Srinivas Ramanathan, "Communication architectures and algorithms for media mixing in multimedia conferences," *IEEE/ACM Transactions on Networking*, February 1993.

H. Schulzrinne, *Issues in designing a transport protocol for audio and video conferences and other multiparticipant real-time applications*, Internet Draft of the Audio-Video Transport Working Group, October 1993.

Clemens Szyperski and Giorgio Ventre, "Efficient group communication with guaranteed quality of service," in *Fourth IEEE Workshop on Future Trends in Distributed Computing Systems*, September 1993.

Clemens Szyperski and Giorgio Ventre, *Efficient multicasting for interactive multimedia applications*, Technical Report TR-93-017, Computer Science Division, University of California at Berkeley, March 1993.

Dinesh C. Verma and P. M. Gopal, "Routing reserved bandwidth multi-point connections," in *ACM SIGCOMM*, September 1993.

Dinesh C. Verma, Hui Zhang, and Domenico Ferrari, "Delay jitter control for real-time communication in a packet switching network," in *Proc. of TriComm 1991*, April 1991.

Harrick M. Vin, P. Venkat Rangan, and Srinivas Ramanathan, "Hierarchical conferencing architectures for inter-group multimedia collaboration," in *ACM Conference on Organizational Computing Systems*, November 1991.

D. Waitzman, C. Partridge, and S. Deering, *Distance vector multicast routing protocol*, Technical Report RFC 1075, Internet, Network Working Group, November 1988.

Bernard M. Waxman, "Routing of multipoint connections," *IEEE Selected Areas in Communications*, December 1988.

Liming Wei and Deborah Estrin, *A comparison of multicast trees and algorithms*, Draft submitted to INFOCOM 1994, 1993.

Garret A. Wollman, notes from Van Jacobson talk of 24 June 1993, presented over MBONE.

S. Zabele and R. Braudes, *Requirements for multicast protocols*, Technical Report RFC 1458, Internet, Network Working Group, May 1993.

Steven Randall Zeswitz, *NPSNET: integration of distributed interactive simulation (DIS) protocol for communication architecture and information interchange*, Master's thesis, Naval Postgraduate School, September 1993.

Hui Zhang and Domenico Ferrari, *Rate-controlled static-priority queueing*, Technical Report TR-92-003, Computer Science Division, University of California at Berkeley, February 1992.

Hui Zhang and Srinivasan Keshav, "Comparison of rate-based service disciplines," in *ACM SIGCOMM*, September 1991.

Lixia Zhang, Stephen E. Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala, "RSVP: a new resource ReSerVation Protocol," *IEEE Networks Magazine*, September 1993.

INITIAL DISTRIBUTION LIST

- | | | |
|----|---|---|
| 1. | Defense Technical Information Center
Cameron Station
Alexandria, VA 22304-6145 | 2 |
| 2. | Dudley Knox Library
Code 52
Naval Postgraduate School
Monterey, CA 93943-5002 | 2 |
| 3. | Chairman, Code EC
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93943 | 1 |
| 4. | Professor Shridhar B. Shukla Code EC/Sh
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93943 | 2 |
| 5. | Professor Gilbert Lundy Code CS/Ln
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943 | 1 |
| 6. | Professor David Pratt Code CS/Pr
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943 | 1 |
| 7. | LT Eric B. Boyer
4965 Sabal Palm Blvd 405
Tamarac, FL 33319 | 1 |